Administración local

Josep Jorba Esteve

P07/M2103/02284



Índice

Introducción

Una de las primeras tareas con la que tendrá que enfrentarse el administrador será la gestión de los recursos locales presentes en la máquina. En el curso de introducción a GNU/Linux, se cubrieron algunos de estos aspectos de forma básica. En el presente, veremos algunas de estas tareas de administración de forma más profunda, y algunos de los aspectos de personalización y rendimiento de los recursos.

Comenzaremos por analizar el proceso de arranque de un sistema GNU/Linux, que nos hará comprender la estructura inicial del sistema y su relación con los diversos servicios que éste proporciona.

A continuación aprenderemos cómo obtener una visión general del estado actual del sistema por medio de los diferentes procedimientos y comandos de que se dispone para evaluar las diversas partes del sistema; de este modo podremos tomar decisiones de administración si detectamos algún fallo o deficiencia de rendimiento, o la falta de algún recurso.

Nota

La administración local engloba muchas tareas variadas, que quizás sean las más utilizadas por el administrador en su trabajo diario.

Uno de los principales puntos de la administración es la gestión de usuarios, ya que cualquier configuración de la máquina estará destinada a que pueda ser utilizada por éstos; veremos cómo definir nuevos usuarios al sistema y controlar su nivel de acceso a los recursos.

En cuanto a los periféricos del sistema, como discos e impresoras, disponemos de diferentes posibilidades de gestión, ya sea vía diferentes servidores (caso impresión) o diferentes sistemas de archivos que podemos tratar, así como algunas técnicas de optimización del rendimiento de los discos.

También examinaremos el problema de la actualización del sistema y cómo mantenerlo actualizado; así como la nueva incorporación de software de aplicación y cómo hacerlo disponible a los usuarios. Asimismo, analizaremos la problemática de ejecutar trabajos temporizados en el sistema.

En el taller final examinaremos la evaluación de estado de una máquina, siguiendo los puntos vistos en este módulo, y llevaremos a cabo algunas de las tareas de administración básicas descritas. En el desarrollo de la unidad comentaremos algunos comandos, y posteriormente, en el taller, veremos algunos de ellos con más detalle en lo que respecta a su funcionamiento y opciones.

1. Distribuciones: particularidades

Intentamos destacar ahora algunas diferencias técnicas menores (que cada vez se reducen más) en las distribuciones (Fedora/Red Hat y Debian) utilizadas [Mor03], que iremos viendo con más detalle a lo largo de las unidades, a medida que vayan apareciendo.

Cambios o particularidades de Fedora/Red Hat:

- Uso del gestor de arranque grub (una utilidad GNU), a diferencia de pasadas versiones de la mayoría de distribuciones que suelen usar lilo, Fedora utiliza grub. GRUB (*grand unified bootloader)* tiene una configuración en modo texto (normalmente en /boot/grub/grub.conf) bastante sencilla, y que puede modificarse en el arranque. Es quizás más flexible que lilo. Últimamente las distribuciones tienden al uso de grub, Debian también lo incluye como opcional.

Nota

Es importante conocer los detalles de una distribución, ya que pueden ser básicos para resolver una tarea o acelerar su solución (por ejemplo, si dispone de herramientas propias).

- Gestión de alternativas. En el caso de que haya más de un software equivalente presente para una tarea concreta, mediante un directorio (/etc/alternatives), se indica cuál es la alternativa que se usa. Este sistema se tomó prestado de Debian, que hace un uso amplio de él en su distribución.

- Programa de escucha de puertos TCP/IP basado en xinetd; en /etc/xinetd.d podemos encontrar de forma modular los ficheros de configuración para algunos de los servicios TCP/IP, junto con el fichero de configuración /etc/xinetd.conf. En los sistemas UNIX clásicos, el programa utilizado es el inetd, que poseía un único fichero de configuración en /etc/inetd.conf, caso, por ejemplo, de la distribución Debian que utiliza inetd, dejando xinetd como opción.

- Algunos directorios de configuración especiales: /etc/profile.d, archivos que se ejecutan cuando un usuario abre un shell; /etc/xinetd.d, configuración de algunos servicios de red; /etc/sysconfig, datos de configuración de varios aspectos del sistema; /etc/cron., varios directorios donde se especifican trabajos para hacer periódicamente (mediante crontab); /etc/pam.d, donde PAM son los denominados módulos de autentificación: en cada uno de cuyos archivos se configuran permisos para el programa o servicio particular; /etc/logrotate.d, configuración de rotación (cuando hay que limpiar, comprimir, etc.) de algunos de los ficheros de log para diferentes servicios.

- Dispone de un software llamado kudzu, que examina el hardware en arranque para detectar posibles cambios de configuración y generar los dispositi-

vos o configuraciones adecuadas. Aunque se está migrando progresivamente a la API Hal que controla precisamente este tema.

En el caso de Debian:

- Sistema de empaquetado propio basado en los paquetes DEB, con herramientas de varios niveles para trabajar con los paquetes como: dpkg, apt-get, dselect, tasksel.

- Debian sigue el FHS, sobre la estructura de directorios, añadiendo algunos particulares en /etc, como por ejemplo: /etc/default, archivos de configuración, y valores por defecto para algunos programas; /etc/network, datos y guiones de configuración de las interfaces de red; /etc/dpkg y /etc/apt, información de la configuración de las herramientas de gestión de paquetes; /etc/alternatives, enlaces a los programas por defecto, en aquellos que hay (o puede haber) varias alternativas disponibles.

- Sistema de configuración de muchos paquetes de software por medio de la herramienta dpkg-reconfigure. Por ejemplo:

```
dpkg-reconfigure gdm
```

  permite escoger el gestor de entrada para X, o:

```
dpkg-reconfigure X-Window-system
```

  nos permite configurar los diferentes elementos de X.

- Utiliza configuración de servicios TCP/IP por inetd, configuración en fichero /etc/inetd.conf; dispone de una herramienta update-inetd para inhabilitar o crear entradas de servicios.

- Algunos directorios de configuración especiales: /etc/cron., varios directorios donde se especifican trabajos para hacer periódicamente (mediante crontab); /etc/pam.d, donde PAM son módulos de autentificación.

2. Niveles de arranque y servicios

Un primer punto importante en el análisis del comportamiento local del sistema, es su funcionamiento en los llamados niveles de ejecución (o runlevels), que determinan (en el nivel) el modo actual de trabajo del sistema, y los servicios que se proporcionan [Wm02].

Un servicio es una funcionalidad proporcionada por la máquina, normalmente basada en *daemonds (*O procesos en segundo plano de ejecución, que controlan peticiones de red, actividad del hardware, u otros programas que provean alguna tarea).

La activación o parada de servicios se realiza mediante la utilización de *scripts*. La mayoría de los servicios estándar, los cuales suelen tener su configuración en el directorio /etc, suelen controlarse mediante los scripts presentes en /etc/init.d/. En este directorio suelen aparecer scripts con nombres similares al servicio donde van destinados, y se suelen aceptar parámetros de activación o parada. Se realiza:

`/etc/init.d/servicio start`	arranque del servicio.
`/etc/init.d/servicio stop`	parada del servicio.
`/etc/init.d/servicio restart`	parada y posterior arranque del servicio.

Cuando un sistema GNU/Linux arranca, primero se carga el kernel del sistema, después se inicia el primer proceso, denominado init, que es el responsable de ejecutar y activar el resto del sistema, mediante la gestión de los niveles de ejecución (o *runlevels*).

> Un nivel de ejecución es básicamente una configuración de programas y servicios que se ejecutarán orientados a un determinado funcionamiento.

Los niveles típicos, aunque puede haber diferencias en el orden, en especial en los niveles 2-5 (en la tabla la configuración en Fedora, y la recomendada por el estándar LSB), suelen ser:

Runlevel	Función	Descripción
0	Parada	Finaliza servicios y programas activos, así como desmonta filesystems activos y para la CPU.
1	Monousuario	Finaliza la mayoría de servicios, permitiendo sólo la entrada del administrador (*root*). Se usa para tareas de mantenimiento y corrección de errores críticos.

Runlevel	Función	Descripción
2	Multiusuario sin red	No se inician servicios de red, permitiendo sólo entradas locales en el sistema.
3	Multiusuario	Inicia todos los servicios excepto los gráficos asociados a X Window.
4	Multiusuario	No suele usarse, típicamente es igual que el 3.
5	Multiusuario X	Igual que el 3, pero con soporte X para la entrada de usuarios (*login* gráfico).
6	Reinicio	Para todos los programas y servicios. Reinicia el sistema.

Por contra cabe señalar que Debian usa un modelo, en el que los niveles 2-5 son prácticamente equivalentes, realizando exactamente la misma función (aunque podría ocurrir que en alguna versión esto fuera a cambiar para coincidir con el estándar LSB).

Estos niveles suelen estar configurados en los sistemas GNU/Linux (y UNIX) por dos sistemas diferentes, el BSD, o el SystemV (a veces abreviado como sysV). En el caso de Fedora y Debian, se utiliza el sistema SystemV, que es el que mostraremos, pero otros UNIX y alguna distribución GNU/Linux (como Slackware) utilizan el modelo BSD.

En el caso del modelo *runlevel* de SystemV, cuando el proceso init arranca, utiliza un fichero de configuración llamado /etc/inittab para decidir el modo de ejecución en el que va a entrar. En este fichero se define el *runlevel* por defecto (*initdefault*) en arranque (por instalación en Fedora el 5, en Debian el 2), y una serie de servicios de terminal por activar para atender la entrada del usuario.

Después, el sistema, según el *runlevel* escogido, consulta los ficheros contenidos en /etc/rcn.d, donde *n* es el número asociado al *runlevel* (nivel escogido), en el que se encuentra una lista de servicios por activar o parar en caso de que arranquemos en el *runlevel,* o lo abandonemos. Dentro del directorio encontraremos una serie de *scripts* o enlaces a los *scripts* que controlan el servicio.

Cada *script* posee un nombre relacionado con el servicio, una S o K inicial que indica si es el *script* para iniciar (S) o matar (K) el servicio, y un número que refleja el orden en que se ejecutarán los servicios.

Una serie de comandos de sistema sirven de ayuda para manejar los niveles de ejecución, cabe mencionar:

- Los scripts, que ya hemos visto, en /etc/init.d/ nos permiten arrancar, parar o reiniciar servicios individuales.

- telinit, nos permite cambiar de nivel de ejecución, sólo tenemos que indicar el número. Por ejemplo, necesitamos hacer una tarea crítica en *root*; sin usuarios trabajando, podemos hacer un telinit 1 (también puede usarse S) para pasar a *runlevel* monousuario, y después de la tarea un telinit 3 para volver a multiusuario. También puede utilizarse el comando init para la misma tarea,

aunque telinit aporta algún parámetro extra. Por ejemplo, el reinicio típico de un sistema UNIX se hacía con sync; sync; sync; init 6, el comando sync fuerza el vaciado de los *buffers* del sistema de archivos, y luego reiniciamos en *runlevel* 6.

- shutdown, permite parar ('h' de *halt*) o reiniciar el sistema ('r' de *reboot*). Puede darse también un intervalo de tiempo para hacerse, o bien inmediatamente. Para estas tareas también existen los comandos halt y reboot.

- wall, permite enviar mensajes de advertencia a los usuarios del sistema. Concretamente, el administrador puede anunciar que se va a parar la máquina en un determinado momento. Comandos como shutdown suele utilizarlosde forma automática.

- pidof, permite averiguar el PID (*process* ID) asociado a un proceso. Con ps obtenemos los listados de procesos, y si queremos eliminar un servicio o proceso mediante kill, necesitaremos su PID.

Respecto a todo el modelo de arranque, las distribuciones presentan algún pequeño cambio:

- Fedora/Red Hat: el runlevel 4 no tiene un uso declarado. Los directorios /etc/rcn.d existen como enlaces hacia subdirectorios de /etc/rc.d, donde están centralizados los *scripts* de arranque. Los directorios son, así: /etc/rc.d/rcn.d; pero como existen los enlaces, es transparente al usuario. El *runlevel* por defecto es el 5 con arranque con X.

  Los comandos y ficheros relacionados con el arranque del sistema están en los paquetes de software sysvinit e initscripts.

  Respecto a los cambios de ficheros y guiones en Fedora, cabe destacar: en /etc/sysconfig podemos encontrar archivos que especifican valores por defecto de la configuración de dispositivos o servicios. El guión /etc/rc.d/rc.sysinit es invocado una vez cuando el sistema arranca; el guión /etc/rc.d/rc.local se invoca al final del proceso de carga y sirve para indicar inicializaciones específicas de la máquina.

  El arranque real de los servicios se hace por medio de los guiones almacenados en /etc/rc.d/init.d. Hay también un enlace desde /etc/init.d. Además, Fedora proporciona unos *scripts* de utilidad para manejar servicios: /sbin/service para parar o iniciar un servicio por el nombre; y /sbin/chkconfig, para añadir enlaces a los ficheros S y K necesarios para un servicio, o la obtención de información sobre los servicios.

- Debian dispone de comandos de gestión de los *runlevels* como update-rc.d, que permite instalar o borrar servicios arrancándolos o parándolos en uno

o más *runlevels*; invoke-rc.d, permite las clásicas acciones de arrancar, parar o reiniciar el servicio.

El *runlevel* por defecto en Debian es el 2, el X Window System no se gestiona desde /etc/inittab, sino que existe el gestor (por ejemplo, gdm o kdm) como si fuera un servicio más del *runlevel* 2.

3. Observar el estado del sistema

Una de las principales tareas del administrador (*root*) en su día a día, será verificar el correcto funcionamiento del sistema y vigilar la existencia de posibles errores o de saturación de los recursos de la máquina (memoria, discos, etc.). Pasaremos a detallar en los siguientes subapartados los métodos básicos para examinar el estado del sistema en un determinado momento y llevar a cabo las acciones necesarias para evitar problemas posteriores.

En el taller final de esta unidad, realizaremos un examen completo de un sistema ejemplo, para que se puedan ver algunas de estas técnicas.

3.1. Arranque del sistema

En el arranque de un sistema GNU/Linux, se produce todo un volcado de información interesante; cuando el sistema arranca, suelen aparecen los datos de detección de las características de la máquina, detección de dispositivos, arranque de servicios de sistema, etc., y se mencionan los problemas aparecidos.

En la mayoría de las distribuciones esto puede verse en la consola del sistema directamente durante el proceso de arranque. Sin embargo, o la velocidad de los mensajes o algunas modernas distribuciones que los ocultan tras carátulas gráficas pueden impedir seguir los mensajes correctamente, con lo que necesitaremos una serie de herramientas para este proceso.

Básicamente, podemos utilizar:

- Comando dmesg: da los mensajes del último arranque del kernel.
- Fichero /var/log/messages: log general del sistema, que contiene los mensajes generados por el kernel y otros daemons (puede haber multitud de archivos diferentes de log, normalmente en /var/log, y dependiendo de la configuración del servicio *syslog*).
- Comando uptime: indica cuánto tiempo hace que el sistema está activo.
- Sistema /proc: pseudo sistema de ficheros (procfs) que utiliza el kernel para almacenar la información de procesos y de sistema.
- Sistema /sys: pseudo sistema de ficheros (sysfs) que apareció con la rama 2.6.x del kernel, con objetivo de proporcionar una forma más coherente de acceder a la información de los dispositivos y sus controladores (*drivers*).

3.2. *Kernel*: directorio /proc

El *kernel* durante su arranque pone en funcionamiento un pseudo-filesystem llamado /proc, donde vuelca la información que recopila de la máquina, así como muchos de sus datos internos, durante la ejecución. El directorio /proc está implementado sobre memoria y no se guarda en disco. Los datos contenidos son tanto de naturaleza estática como dinámica (varían durante la ejecución).

Hay que tener en cuenta que al ser /proc fuertemente dependiente del *kernel*, propicia que su estructura dependa del *kernel* de que dispone el sistema y la estructura y los ficheros incluidos pueden cambiar.

Una de las características interesantes es que en el directorio /proc podremos encontrar las imágenes de los procesos en ejecución, junto con la información que el *kernel* maneja acerca de ellos. Cada proceso del sistema se puede encontrar en el directorio /proc/<pidproceso>, donde hay un directorio con ficheros que representan su estado. Esta información es básica para programas de depuración, o bien para los propios comandos del sistema como *ps* o *top,* que pueden utilizarla para ver el estado de los procesos. En general muchas de las utilidades del sistema consultan la información dinámica del sistema desde /proc (en especial algunas utilidades proporcionadas en el paquete procps).

Nota

El directorio /proc es un recurso extraordinario para obtener información de bajo nivel sobre el funcionamiento del sistema, muchos comandos de sistema se apoyan en él para sus tareas.

Por otra parte, en /proc podemos encontrar otros ficheros de estado global del sistema. Comentamos brevemente algunos ficheros que podremos examinar para obtener información importante:

Fichero	Descripción
/proc/bus	Directorio con información de los buses PCI y USB
/proc/cmdline	Línea de arranque del *kernel*
/proc/cpuinfo	Información de la CPU
/proc/devices	Listado de dispositivos del sistema de caracteres o bloques
/proc/drive	Información de algunos módulos *kernel* de hardware
/proc/filesystems	Sistemas de ficheros habilitados en el *kernel*
/proc/ide	Directorio de información del bus IDE, características de discos
/proc/interrups	Mapa de interrupciones hardware (IRQ) utilizadas
/proc/ioports	Puertos de E/S utilizados
/proc/meminfo	Datos del uso de la memoria
/proc/modules	Módulos del *kernel*
/proc/mounts	Sistemas de archivos montados actualmente
/proc/net	Directorio con toda la información de red
/proc/scsi	Directorio de dispositivos SCSI, o IDE emulados por SCSI
/proc/sys	Acceso a parámetros del *kernel* configurables dinámicamente
/proc/version	Versión y fecha del *kernel*

A partir de la rama 2.6 del *kernel*, se ha iniciado una transición progresiva de procfs (/proc) a sysfs (/sys) con objetivo de mover toda aquella información que no esté relacionada con procesos, en especial dispositivos y sus controladores (módulos del *kernel*) hacia el sistema /sys.

3.3. *Kernel*: /sys

El sistema Sys se encarga de hacer disponible la información de dispositivos y controladores, información de la cual dispone el *kernel*, al espacio de usuario de manera que otras API o aplicaciones puedan acceder de una forma flexible a la información de los dispositivos (u sus controladores). Suele ser utilizada por capas como HAL y el servicio udev para la monitorización y configuración dinámica de los dispositivos.

Dentro del concepto de sys existe una estructura de datos en árbol de los dispositivos y controladores (digamos el modelo conceptual fijo), y cómo después se accede a él a través del sistema de ficheros sysfs (la estructura del cual puede cambiar entre versiones).

En cuanto se detecta o aparece en el sistema un objeto añadido, en el árbol del modelo de controladores (controladores, dispositivos incluyendo sus diferentes clases) se crea un directorio en sysfs. La relación padre/hijo se refleja con subdirectorios bajo /sys/devices/ (reflejando la capa física y sus identificadores). En el subdirectorio /sys/bus se colocan enlaces simbólicos, reflejando el modo en el que los dispositivos pertenecen a los diferentes buses físicos del sistema. Y en /sys/class muestra los dispositivos agrupados de acuerdo a su clase, como por ejemplo red, mientras que /sys/block/ contiene los dispositivos de bloques.

Alguna de la información proporcionada por /sys puede encontrarse también en /proc, pero se consideró que éste estaba mezclando diferentes cosas (dispositivos, procesos, datos hardware, parámetros *kernel*) de forma no coherente, y ésta fue una de las decisiones para crear /sys. Se espera que progresivamente se migre información de /proc a /sys para centralizar la información de los dispositivos.

3.4. Procesos

Los procesos que se encuentren en ejecución en un determinado momento serán, en general, de diferente naturaleza. Podemos encontrar:

- **Procesos de sistema**, ya sean procesos asociados al funcionamiento local de la máquina, *kernel*, o bien procesos (denominados *daemons*) asociados al control de diferentes servicios. Por otro lado pueden ser locales, o de red,

depende si se está ofreciendo el servicio (actuamos de servidor) o estamos recibiendo los resultados del servicio (actuamos de clientes). La mayoría de estos procesos aparecerán asociados al usuario *root*, aunque no estemos presentes en ese momento como usuarios. Puede haber algunos servicios asociados a otros usuarios de sistema (lp, bin, www, mail, etc.), estos son usuarios "virtuales", no interactivos, que utiliza el sistema para ejecutar ciertos procesos.

- **Procesos del usuario administrador**: en caso de actuar como *root*, nuestros procesos interactivos o aplicaciones lanzadas también aparecerán como procesos asociados al usuario *root*.

- **Procesos de usuarios del sistema**: asociados a la ejecución de sus aplicaciones, ya sea tareas interactivas en modo texto o en modo gráfico.

Como comandos rápidos y más útiles podemos utilizar:

- ps: el comando estándar, lista los procesos con sus datos de usuario, tiempo, identificador de proceso y línea de comandos usada. Una de las opciones más utilizada es *ps -ef* (o *-ax*), pero hay muchas opciones disponibles (ver man).

- top*:* una versión que nos da una lista actualizada a intervalos, monitorizando dinámicamente los cambios. Y nos permite ordenar el listado de procesos, por diferentes *ítems*, como gasto de memoria, de uso CPU, con propósito de obtener un *ranking* de los procesos que acaparan los recursos. Muy útil para dar indicios en situaciones extremas de saturación de uso de recursos, de la posible fuente de problemas.

- kill*:* nos permite eliminar procesos del sistema mediante el envío de señales al proceso como, por ejemplo, la de terminación kill -9 pid_del_proceso (9 corresponde a SIGKILL*),* donde indicamos el identificador del proceso. Útil para procesos con comportamiento inestable o programas interactivos que han dejado de responder. Podemos ver una lista de las señales validas en el sistema con man 7 signal

3.5. *Logs* del sistema

Tanto el *kernel* como muchos de los *daemons* de servicios, así como diferentes aplicaciones o subsistemas de GNU/Linux, pueden generar mensajes que vayan a parar a ficheros *log*, ya sea para tener una traza de su funcionamiento, o bien para detectar errores o advertencias de malfuncionamiento o situaciones críticas. Este tipo de *logs* son imprescindibles en muchos casos para las tareas de administración y se suele emplear bastante tiempo de administración en el procesamiento y análisis de sus contenidos.

La mayor parte de los *logs* se generan en el directorio /var/log, aunque algunas aplicaciones pueden modificar este comportamiento; la mayoría de *logs* del propio sistema sí que se encuentran en este directorio.

Un *daemon* particular del sistema (importante) es el daemon *Syslogd*. Este *daemon* se encarga de recibir los mensajes que se envían por parte del *kernel* y otros *daemons* de servicios y los envía a un fichero log que se encuentra en /var/log/messages. Éste es el fichero por defecto, pero Syslogd es también configurable (en el fichero /etc/syslog.conf), de manera que se pueden generar otros ficheros dependiendo de la fuente, según el *daemon* que envía el mensaje, y así dirigirlo a un *log* u a otro (clasificando así por fuente), y/o también clasificar los mensajes por importancia (nivel de prioridad): *alarm, warning, error, critical*, etc.

Nota

El daemon Syslogd es el servicio más importante de obtención de información dinámica de la máquina. El proceso de análisis de los logs nos ayuda a entender el funcionamiento, los posibles errores y el rendimiento del sistema.

Dependiendo de la distribución, puede estar configurado de diferentes modos por defecto, en /var/log suele generar (por ejemplo) en Debian ficheros como: *kern.log, mail.err, mail.info*... que son los *logs* de diferentes servicios. Podemos examinar la configuración para determinar de dónde provienen los mensajes y en qué ficheros los guarda. Una opción que suele ser útil es la posibilidad de enviar los mensajes a una consola virtual de texto (en /etc/syslog.conf se especifica para el tipo, o tipos, de mensaje una consola de destino, como /dev/tty8 o /dev/xconsole), de manera que podremos ir viendo los mensajes a medida que se produzcan. Esto suele ser útil para monitorizar la ejecución del sistema sin tener que estar mirando los ficheros de log a cada momento. Una modificación simple de este método podría ser introducir, desde un terminal, la instrucción siguiente (para el log general):

```
tail -f /var/log/messages
```

Esta sentencia nos permite dejar el terminal o ventana de terminal, de manera que irán apareciendo los cambios que se produzcan en el fichero.

Otros comandos relacionados:

- uptime: tiempo que hace que el sistema está activo. Útil para comprobar que no hay existido algún rearranque del sistema inesperado.

- last: analiza log de entradas/salidas del sistema (/var/log/wtmp) de los usuarios, y los rearranques del sistema. O lastlog control de la última vez que los usuarios han sido vistos en el sistema (información en /var/log/lastlog).

- Varias utilidades para procesamiento combinado de *logs,* que emiten resúmenes (o alarmas) de lo sucedido en el sistema, como por ejemplo: logwatch, logcheck(Debian), log_analysis (Debian)...

3.6. Memoria

Respecto a la memoria del sistema, tendremos que tener en cuenta que disponemos: a) de la memoria física de la propia máquina, b) memoria virtual, que puede ser direccionada por los procesos. Normalmente (a no ser que estemos tratando con servidores empresariales), no dispondremos de cantidades demasiado grandes, de modo que la memoria física será menor que el tamaño de memoria virtual necesario (4GB en sistemas de 32bits). Esto obligará a utilizar una zona de intercambio (*swap*) sobre disco, para implementar los procesos asociados a memoria virtual.

Esta zona de intercambio (*swap*) puede implementarse como un fichero en el sistema de archivos, pero es más habitual encontrarla como una partición de intercambio (llamada de *swap*), creada durante la instalación del sistema. En el momento de particionar el disco, se declara como de tipo Linux Swap.

Para examinar la información sobre memoria tenemos varios métodos y comandos útiles:

- Fichero /etc/fstab: aparece la partición swap (si existiese). Con un comando fdisk podemos averiguar su tamaño (o consulta a /proc/swaps).

- Comando ps*:* permite conocer qué procesos tenemos, y con las opciones de porcentaje y memoria usada.

- Comando top*:* es una versión ps dinámica actualizable por periodos de tiempo. Puede clasificar los procesos según la memoria que usan o el tiempo de CPU.

- Comando free*:* informa sobre el estado global de la memoria. Da también el tamaño de la memoria virtual.

- Comando vmstat*:* informa sobre el estado de la memoria virtual, y el uso que se le da.

- Algunos paquetes como dstat permite recoger datos de los diferentes parámetros (memoria, swap, y otros) a intervalos de tiempo (de forma parecida a top).

3.7. Discos y *filesystems*

Examinaremos qué discos tenemos disponibles, cómo están organizados y de qué particiones y archivos de sistemas (*filesystems*) disponemos.

Cuando dispongamos de una partición, y dispongamos de un determinado *filesystem* accesible, tendremos que realizar un proceso de montaje, para inte-

grarla en el sistema, ya sea explícitamente o bien programada en arranque. En el proceso de montaje se conecta el sistema de archivos asociado a la partición a un punto del árbol de directorios.

Para conocer los discos (o dispositivos de almacenamiento) que tenemos en el sistema, podemos basarnos en la información de arranque del sistema (dmesg), donde se detectan los presentes, como los /dev/hdx para los dispositivos IDE o los SCSI con dispositivos /dev/sdx. Otros dispositivos, como discos duros conectados por USB, discos flash (los de tipo *pen drive*), unidades removibles, CD-ROM externos, suelen ser dispositivos con algún tipo de emulación SCSI, por lo que se verán como dispositivo de este tipo.

Cualquier dispositivo de almacenamiento presentará una serie de particiones de su espacio. Típicamente, un disco IDE soporta un máximo de cuatro particiones físicas, o más si éstas son lógicas (que permiten colocar varias particiones de este tipo sobre una física). Cada partición puede contener tipos de *filesystems* diferentes, ya sean de un mismo operativo o de operativos diferentes.

Para examinar la estructura de un dispositivo conocido, o cambiar su estructura particionando el disco, podemos utilizar el comando fdisk, o cualquiera de sus variantes más o menos interactivas (cfdisk, sfdisk). Por ejemplo, al examinar un disco ejemplo ide /dev/hda, nos da la siguiente información:

```
# fdisk /dev/hda        (dentro opción p)
Disk /dev/hda: 20.5 GB, 20520493056 bytes 255 heads, 63
sectors/track, 2494 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device       Boot    Start   End     Blocks     Id System
/dev/hda1    *       1       1305    10482381   7 HPFS/NTFS
/dev/hda2    *       1306    2429    9028530    83 Linux
/dev/hda3            2430    2494    522112+    82 Linux swap
```

Disco de 20 GB con tres particiones (se identifican con el número añadido al nombre del dispositivo), donde observamos dos particiones con arranque (columna Boot con *) de tipo NTFS y Linux, lo que supone la existencia de un Windows NT/2000/XP/Vista junto con una distribución GNU/Linux, y la última partición que es usada de swap para Linux. Además tenemos información de la estructura del disco y de los tamaños de cada partición.

De los discos y particiones de que dispongamos, algunos de ellos se encontrarán montados en nuestro sistema de ficheros, o estarán preparados para montarse bajo demanda, o bien montarse en el momento en que se disponga de medio (en el caso de dispositivos extraíbles).

Esta información la podemos obtener de diferentes maneras (veremos más detalle en el taller final):

- Fichero /etc/fstab. Indica dispositivos que están preparados para montarse en el arranque o los extraíbles que podrán ser montados. No tienen por qué estar todos los del sistema, sino sólo aquellos que queramos tener en arranque. Los demás podemos montarlos bajo demanda con el comando mount, o desmontarlos con umount.

- Comando mount. Nos informa de los *filesystems* montados en ese momento (ya sean dispositivos reales o *filesystem* virtuales como /proc). Podemos obtener esta información también desde el fichero /etc/mtab.

- Comando df -k. Nos informa de los *filesystems* de almacenamiento, y nos permite verificar el espacio usado y disponible. Comando básico para controlar espacio de disco disponible.

Respecto a este ultimo comando df -k, una de nuestras tareas básicas de administración de la máquina es controlar los recursos de la máquina, y en este caso el espacio disponible en los *filesystems* utilizados. Estos tamaños hay que monitorizarlos con cierta frecuencia para evitar la caída del sistema; nunca tendría que dejarse un *filesystem* (y más si es el /) por debajo de un 10 o 15%, ya que hay muchos procesos *daemons* que están escribiendo normalmente información temporal o *logs,* que pueden generar gran información; un caso particular lo forman los ficheros core, ya comentados, que pueden suponer (dependiendo del proceso) tamaños muy grandes de archivo. Normalmente, habrá que tener algunas precauciones de "limpieza del sistema" si se detectan situaciones de saturación del *filesystem*:

- Eliminar temporales antiguos. Los directorios /tmp y /var/tmp suelen acumular muchos archivos generados por diferentes usuarios o aplicaciones. Algunos sistemas o distribuciones ya toman medidas de limpieza, como limpiar /tmp en cada arranque del sistema.

- Logs: evitar su crecimiento excesivo, según la configuración del sistema (por ejemplo de Syslogd) la información generada de mensajes puede ser muy grande. Normalmente, habrá que limpiar periódicamente al llegar a determinados tamaños, y en todo caso, si necesitamos la información para posteriores análisis, podemos realizar *backups* en medios extraíbles. Este proceso puede automatizarse mediante uso de *scripts* cron, o bien por medio de herramientas especializadas como logrotate.

- Hay otros puntos del sistema que suelen crecer mucho, como pueden ser: a) ficheros core de los usuarios: podemos eliminarlos periódicamente o eliminar su generación; b) el sistema de correo electrónico: almacena todos los correos enviados y recibidos, podemos pedir a los usuarios que hagan

limpieza periódica, o bien poner sistemas de cuotas; c) las cachés de los navegadores u otras aplicaciones: también suelen tener tamaños grandes, otra limpieza que habrá que hacer periódicamente; d) las cuentas de los propios usuarios: pueden tener cuotas para no exceder los tamaños prefijados, etc.

4. Sistema de ficheros

En cada máquina con un sistema GNU/Linux podemos encontrarnos con diferentes sistemas de ficheros de diferentes tipos [Hin].

Para empezar, es habitual encontrarse con los propios sistemas de ficheros Linux creados en varias particiones de los discos [Koe]. La configuración habitual suele ser de dos particiones: la correspondiente a "/" (*root filesystem*) y la correspondiente al fichero de intercambio o de swap. Aunque en configuraciones más profesionales, suele ser habitual, separar particiones con partes "diferenciadas" del sistema, una técnica habitual es, por ejemplo (veremos más opciones después), crear particiones diferentes para:

/ /boot /home /opt /tmp /usr /var swap

Que seguramente se encontrarán montadas desde diferentes orígenes (diferentes discos, o incluso red en algunos casos). La idea es separar claramente partes estáticas y dinámicas del sistema, como permitir de una forma más fácil, ante problemas de saturación, extender las particiones. O aislar más fácilmente partes para la realización de *backups* (por ejemplo las cuentas de los usuarios en la partición /home).

El tipo de particiones swap es de tipo Linux swap, y la correspondiente a / suele ser de alguno de los sistemas de ficheros estándar, ya sea ext2 (el tipo por defecto hasta los *kernels* 2.4), o el nuevo ext3, que es una mejora del ext2 compatible pero con *journaling*, lo que permite tener un log de lo que va pasando al sistema de ficheros, para recuperaciones más rápidas en caso de error. También pueden ser habituales otros sistemas de archivos como Reiser o XFS.

Otra configuración habitual puede ser de tres particiones: /, swap, /home, donde la /home se dedicará a las cuentas de los usuarios. Esto permite separar las cuentas de los usuarios del sistema, aislando en dos particiones separadas, y podemos dar el espacio necesario para las cuentas en otra partición.

Otro esquema muy utilizado es el de separar en particiones las partes estáticas del sistema de las dinámicas, por ejemplo, una partición donde se coloca / con la parte estática (/bin /sbin y /usr en algunos casos) que se espera que no va a crecer o lo va a hacer muy poco, y otra o varias con la parte dinámica (/var /tmp /opt), suponiendo que /opt, por ejemplo, es el punto de instalación del software nuevo. Esto permite ajustar mejor el espacio de disco y dejar más espacio para las partes del sistema que lo necesiten.

Respecto a los sistemas de ficheros soportados debemos destacar la gran variedad de ellos, actualmente podemos encontrar (entre otros):

- Sistemas asociados a GNU/Linux, como el estandard ext2, y el ext3, evolución del anterior con concepto de *journaling* (soporte de log de operaciones realizadas en el sistema de fichero que puede permitir su recuperación en caso de algún desastre que lo haga inconsistente).

- Compatibilidad con entornos no GNU/Linux: msdos, vfat, ntfs, acceso a los diferentes sistemas de fat16, fat32, y ntfs. En particular resaltar que el soporte *kernel*, en el caso del kernel esta limitado a lectura. Pero como ya hemos comentado, existen soluciones en espacio de usuario (mediante FUSE, un componente que permite escribir sistemas de ficheros en espacio de usuario), que la permiten, como el ya mencionado ntfs-3g. También se disponen de compatibilidad a otros entornos como Mac con hfs y hfsplus.

- Sistems asociados a soportes físicos, como CD/DVD como los iso9660, y udf.

- Sistemas usados en diferentes Unix, que ofrecen generalmente mejor rendimiento (a veces a costa de mayor consumo de recursos, en CPU por ejemplo), como JFS2 (IBM), XFS (SGI), o ReiserFS.

- Sistemas de ficheros en red (más tradicionales): NFS, Samba (smbfs, cifs), permiten acceder a sistemas de ficheros disponibles en otras máquinas de forma transparente por red.

- Sistemas distribuidos en red: como GFS, Coda.

- Pseudo Sistemas de ficheros, como procfs (/proc) o sysfs (/sys).

Nota

El documento Filesystems Howto, da breves explicaciones de los diversos sistemas de ficheros así como direcciones web de interés para cada uno de ellos.

En la mayoría (excepto algún caso especial) de estos sistemas de ficheros, GNU/Linux, nos permitirá crear particiones de estos tipos, construir el sistema de ficheros del tipo requerido, y montarlas como parte integrante del árbol de directorios, ya sea de forma temporal o permanente.

4.1. Puntos de montaje

Aparte del *filesystem* principal / y de sus posibles divisiones en particiones extras (/usr /var /tmp /home), cabe tener en cuenta la posibilidad de dejar puntos de montaje preparados para el montaje de otros sistemas de ficheros, ya sea particiones de disco u otros dispositivos de almacenamiento.

En las máquinas en que GNU/Linux comparte la partición con otros sistemas operativos, mediante algún sistema de arranque (lilo o grub), pueden existir varias particiones asignadas a los diferentes operativos. Muchas veces es inte-

resante compartir datos con estos sistemas, ya sea para leer sus ficheros o modificarlos. A diferencia de otros sistemas (que sólo tienen en cuenta sus propios datos y sistemas de ficheros, y en los cuales en algunas versiones no se soportan algunos de sus propios sistemas de ficheros), GNU/Linux es capaz de tratar, como hemos visto, con una cantidad enorme de sistemas de ficheros de diferentes operativos y poder compartir la información.

Ejemplo

Si en los PC personales hemos instalado GNU/Linux, seguramente encontraremos más de un operativo, por ejemplo, otra versión de GNU/Linux con ext2 o 3 de sistema de ficheros, podríamos encontrar un antiguo msdos con su sistema de ficheros FAT, un Windows98/ME/XP Home con FAT32 (o vfat para Linux), o un Windows NT/2000/XP/Vista con sistemas NTFS (ntfs para Linux) y FAT32 (vfat) a la vez.

Nuestro sistema GNU/Linux puede leer datos (o sea ficheros y directorios) de todos estos sistemas de ficheros y escribir en la mayoría de ellos.

En el caso de NTFS, hasta ciertos momentos, existieron problemas en la escritura que estaba en forma experimental en la mayoría de *drivers* de *kernel* aparecidos. Debido principalmente a las diferentes versiones que van apareciendo del sistema de ficheros, ya que existen dos versiones principales llamadas NTFS y NTFS2, y algunas extensiones como los llamados volúmenes dinámicos, o los sistemas de ficheros encriptados. Y acceder con según que *drivers* presentaba ciertas incompatibilidades, que podrían causar corrupciones de datos o errores en el sistema de ficheros.

Debido a FUSE, un módulo integrado en el *kernel* (a partir de 2.6.11), se ha permitido un desarrollo más flexible de sistemas de ficheros, directamente en espacio de usuario (de hecho FUSE actúa como un "puente" entre las peticiones del *kernel*, y el acceso que se hace desde el *driver*).

Gracias a las posibilidades de FUSE, se tiene un soporte más o menos completo de NTFS, (mientras Microsoft no haga más cambios en la especificación), en especial desde la aparición del *driver* (basado en FUSE) ntfs-3g (http://www.ntfs-3g.org), y la combinación con las utilidades ntfsprogs.

Para que se puedan leer o escribir los datos, la partición tiene que estar disponible dentro de nuestro sistema de ficheros raíz (/). Por lo tanto, hay que llevar a cabo un proceso de "montaje" del sistema de ficheros en algún punto de nuestro árbol de directorios. Se seguirá el mismo proceso si se trata de un dispositivo de almacenamiento, ya sea disquete o floppy.

Dependiendo de la distribución, se usan unos u otros, o también los podemos crear nosotros. Normalmente suelen existir o bien como subdirectorios de la raíz, por ejemplo /cdrom /win /floppy, o bien son subdirectorios dentro de /mnt, el punto estándar de montaje (aparecen como /mnt/cdrom /mnt/floppy...), o el directorio /media preferido últimamente por las distribuciones. Según el es-

tandard FHS, /mnt se debería usar para montajes temporales de sistemas de archivo, mientras /media se utilizaría para montar dispositivos removibles.

El proceso de montaje se realiza mediante la orden mount con el siguiente formato:

```
mount -t filesystem-type device mount-point
```

El tipo de *filesystem* puede ser: msdos (fat), vfat (fat32), ntfs (ntfs lectura), iso9660 (para cdrom)... (de los posibles).

El dispositivo es la entrada correspondiente en el directorio /dev a la localización del dispositivo, los IDE tenían /dev/hdxy donde x es a,b,c, o d (1 master, 1 slave, 2 master, 2 slave) e y, el número de partición, los SCSI (/dev/sdx) donde x es a,b,c,d ... (según el ID SCSI asociado 0,1,2,3,4 ...).

Vamos a ver algunos ejemplos:

```
mount -t iso9660 /dev/hdc /mnt/cdrom
```

montaría el CD-ROM (si es el IDE que está en el segundo IDE de forma máster) en el punto /mnt/cdrom.

```
mount -t iso9660 /dev/cdrom /mnt/cdrom
```

montaría el CD-ROM; /dev/cdrom se usa como sinónimo (es un link) del dispositivo donde está conectado.

```
mount -t vfat /dev/fd0H1440 /mnt/floppy
```

montaría el disquete, /dev/fd0H1440. Sería la disquetera A en alta densidad (1.44 MB), también puede usarse /dev/fd0.

```
mount -t ntfs /dev/hda2 /mnt/winXP
```

montaría la segunda partición del primer dispositivo IDE (la C:), de tipo NTFS (por ejemplo un Windows XP).

Si estas particiones son más o menos estables en el sistema (o sea, no cambian frecuentemente) y las queremos utilizar, lo mejor será incluir los montajes para que se hagan en tiempo de ejecución, al iniciar el sistema, mediante la configuración del fichero /etc/fstab:

```
# /etc/fstab: Información estática del sistema de ficheros
#
#<Sis. ficheros><Punto montaje><Tipo><Opciones>
<volcado><pasada>
```

```
/dev/hda2    /          ext3     errors = remountro  0  1
/dev/hdb3    none       swap     sw                  0  0
proc         /proc      proc     defaults            0  0
/dev/fd0     /floppy    auto     user,noauto         0  0
/dev/cdrom   /cdrom     iso9660  ro,user,noauto      0  0
/dev/sdb1    /mnt/usb   vfat     user,noauto         0  0
```

Por ejemplo, esta configuración incluye algunos de los sistemas estándar, como la raíz en /dev/hda2, la partición de swap que está en hdb3, el sistema proc (que utiliza el *kernel* para guardar su información). Y el disquete, el CD-ROM, y en este caso un disco USB de tipo Flash (que se detecta como un dispositivo SCSI). En algunos casos, se especifica auto como tipo de *filesystem*. Esto permite que se autodetecte el sistema de ficheros. Si se conoce, es mejor indicarlo en la configuración, y por otra parte, el noauto en las opciones permite que no sea montado de forma automática siempre, sino bajo petición (o acceso).

Si tenemos esta información en el fichero, el proceso de montaje se simplifica mucho, ya que se hará o bien en ejecución, en arranque, o bien bajo demanda (para los noauto). Y puede hacerse ahora simplemente pidiendo que se monte el punto de montaje o el dispositivo:

```
mount /mnt/cdrom
mount /dev/fd0
```

dado que el sistema ya tiene el resto de la información.

El proceso contrario, el desmontaje, es bastante sencillo, el comando umount con punto o dispositivo:

```
umount /mnt/cdrom
umount /dev/fd0
```

En el caso de medios extraíbles, tipo CD-ROM (u otros), puede usarse eject para la extracción del soporte físico:

```
eject /dev/cdrom
```

o, en este caso, sólo:

```
eject
```

Los comandos mount y umount montan o desmontan todos los sistemas disponibles. En el fichero */etc/mtab* se mantiene una lista de los sistemas montados en un momento concreto, que se puede consultar, o ejecutar mount sin parámetros para obtener esta información.

4.2. Permisos

Otro tema que habrá que controlar en el caso de los ficheros y directorios es el de los permisos que queremos establecer en cada uno de ellos, recordando que cada fichero puede disponer de la serie de permisos: rwxrwxrwx donde se corresponden con rwx del propietario, rwx del grupo al que el usuario pertenece, y rwx para otros usuarios. En cada uno se puede establecer el permiso de lectura (r), escritura (w) o ejecución (x). En el caso de un directorio, x denota el permiso para poder entrar en ese directorio (con el comando cd, por ejemplo).

Para modificar los derechos sobre un directorio o fichero, existen los comandos:

- chown: cambiar propietario de los ficheros.
- chgrp: cambiar grupo propietario de los ficheros.
- chmod: cambiar permisos específicos (rwx) de los archivos.

Estos comandos también permiten la opción -R, que es recursiva si se trata de un directorio.

5. Usuarios y grupos

Los usuarios de un sistema GNU/Linux disponen normalmente de una cuenta asociada (definida con algunos de sus datos y preferencias), junto con el espacio en disco para que puedan desarrollar sus archivos y directorios. Este espacio está asignado al usuario, y sólo puede ser usado por éste (a no ser que los permisos especifiquen cosas diferentes).

Dentro de las cuentas asociadas a usuarios podemos encontrar diferentes tipos:

- La del administrador, con identificador *root*, que sólo es (o debería ser) utilizada para las operaciones de administración. El usuario *root* es el que dispone de más permisos y acceso completo a la máquina y a los archivos de configuración. Por lo tanto, también es el que más daño puede causar por errores u omisiones. Es mejor evitar usar la cuenta de *root* como si fuese un usuario más, por lo que se recomienda dejarla sólo para operaciones de administración.

- Cuentas de usuarios: las cuentas normales para cualquier usuario de la máquina tienen los permisos restringidos al uso de ficheros de su cuenta, y a algunas otras zonas particulares (por ejemplo, los temporales en /tmp), así como a utilizar algunos dispositivos para los que se le haya habilitado.

- Cuentas especiales de los servicios: lp, news, wheel, www-data... cuentas que no son usadas por personas, sino por servicios internos del sistema, que los usa bajo estos nombres de usuario. Algunos de los servicios también son usados bajo el nombre de *root*.

Un usuario normalmente se crea mediante la especificación de un nombre (o identificador de usuario), una palabra de paso (*password*) y un directorio personal asociado (la cuenta).

La información de los usuarios del sistema está incluida en los siguientes archivos:

```
/etc/passwd
/etc/shadow
/etc/group
/etc/gshadow
```

Ejemplo de unas líneas del /etc/passwd:

```
juan:x:1000:1000:Juan Garcia,,,:/home/juan:/bin/bash
root:x:0:0:root:/root:/bin/bash
```

donde se indica (si aparecen :: seguidos, que el campo está vacío):

- juan: identificador de usuario en el sistema.
- x: palabra de paso del usuario codificada, si hay una “x” es que se encuentra en el fichero /etc/shadow.
- 1000: código del usuario, lo usa el sistema como código de identidad del usuario.
- 1000: código del grupo principal al que pertenece, la información del grupo en /etc/group.
- Juan García: comentario, suele ponerse el nombre completo del usuario.
- /home/juan: directorio personal asociado a su cuenta.
- /bin/bash: *shell* interactivo que utilizará el usuario al interactuar con el sistema, en modo texto, o por *shell* gráfico. En este caso, el shell Bash de GNU, que es el utilizado por defecto. El fichero /etc/passwd solía contener las palabras de paso de los usuarios de forma encriptada, pero el problema estaba en que cualquier usuario podía ver el fichero, y en su momento se diseñaron cracks que intentaban encontrar en forma bruta la palabra de paso, mediante la palabra de paso encriptada como punto de partida (palabra codificada con el sistema *crypt).*

Para evitar esto, hoy en día ya no se colocan las palabras de paso en este archivo, sólo una “x” que indica que se encuentran en otro fichero, que es sólo de lectura para el usuario *root*, /etc/shadow, cuyo contenido podría ser algo parecido a lo siguiente:

```
juan:a1gNcs82ICst8CjVJS7ZFCVnu0N2pBcn/:12208:0:99999:7:::
```

donde se encuentra el identificador del usuario junto con la palabra de paso encriptada. Además, aparecen como campos separados por “:”:

- Días desde 1 de enero de 1970 en que la palabra de paso se cambió por última vez.
- Días que faltan para que se cambie (0 no hay que cambiarla).
- Días después en que hay que cambiarla (o sea, plazo de cambio).
- Días en que el usuario será avisado antes de que le expire.
- Días una vez expirado, en que se producirá la deshabilitación de la cuenta.

- Días desde 1 enero 1970 en que la cuenta está deshabilitada.

- Y un campo reservado.

Además, las claves de encriptación pueden ser más difíciles, ya que ahora puede utilizarse un sistema denominado md5 (suele aparecer como opción a la hora de instalar el sistema) para proteger las palabras de paso de los usuarios. Veremos más detalles al respecto en la unidad dedicada a la seguridad.

En /etc/group está la información de los grupos de usuarios:

```
jose:x:1000:
```

donde tenemos:

```
nombre-grupo:contraseña-grupo:identificador-del-
grupo:lista-usuarios
```

La lista de usuarios del grupo puede estar presente o no, ya que la información ya está en /etc/passwd, no suele ponerse en /etc/group. Si se pone, suele aparecer como una lista de usuarios separada por comas. Los grupos también pueden poseer una contraseña asociada (aunque no suele ser tan común), como en el caso de los de usuario, también existe un fichero de tipo *shadow*: /etc/gshadow.

Otros ficheros interesantes son los del directorio /etc/skel, donde se hallan los ficheros que se incluyen en cada cuenta de usuario al crearla. Recordar que, como vimos con los *shell* interactivos, podíamos tener unos *scripts* de configuración que se ejecutaban al entrar o salir de la cuenta. En el directorio skel se guardan los “esqueletos” que se copian en cada usuario al crearlo. Suele ser responsabilidad del administrador crear unos ficheros adecuados para los usuarios, poniendo los *path* necesarios de ejecución, inicialización de variables de sistema, variables que se necesiten para el software, etc.

A continuación vamos a ver una serie de comandos útiles para esta administración de usuarios (mencionamos su funcionalidad y en el taller haremos algunas pruebas):

- useradd: añadir un usuario al sistema.

- userdel*:* borrar un usuario del sistema.

- usermod*:* modificar un usuario del sistema.

- groupadd, groupdel, groupmod lo mismo para grupos.

- newusers, chpasswd: pueden ser de utilidad en grandes instalaciones con muchos usuarios, ya que permiten crear varias cuentas desde la informa-

ción introducida en un fichero (*newusers)* o bien cambiar las contraseñas a un gran número de usuarios (*chpasswd).*

- chsh*:* cambiar el *shell* de *login* del usuario.

- chfn*:* cambiar la información del usuario, la presente en el comentario del fichero /etc/passwd.

- passwd*:* cambia la contraseña de un usuario. Puede ejecutarse como usuario, y entonces pide la contraseña antigua y la nueva. En el caso de hacerlo, el *root* tiene que especificar el usuario al que va a cambiar la contraseña (si no, estaría cambiando la suya) y no necesita la contraseña antigua. Es quizás el comando más usado por el *root*, de cara a los usuarios cuando se les olvida la contraseña antigua.

- su*:* una especie de cambio de identidad. Lo utilizan tanto usuarios, como el *root* para cambiar el usuario actual. En el caso del administrador, es bastante utilizado para testear que la cuenta del usuario funcione correctamente; hay diferentes variantes: su (sin parámetros, sirve para pasar a usuario *root*, previa identificación, permitiendo que cuando estamos en una cuenta de usuario, pasar a *root* para hacer alguna tarea). La sentencia su iduser (cambia el usuario a iduser, pero dejando el entorno como está, o sea, en el mismo directorio...). El mandato su - iduser (hace una sustitución total, como si el segundo usuario hubiese entrado en el sistema haciendo un *login*).

Respecto a la administración de usuarios y grupos, lo que hemos comentado aquí hace referencia a la administración local de una sola máquina. En sistemas con múltiples máquinas que comparten los usuarios suele utilizarse otro sistema de gestión de la información de los usuarios. Estos sistemas, denominados genéricamente sistemas de información de red, como NIS, NIS+ o LDAP, utilizan bases de datos para almacenar la información de los usuarios y grupos, de manera que se utilizan máquinas servidoras, donde se almacena la base de datos, y otras máquinas clientes, donde se consulta esta información. Esto permite tener una sola copia de los datos de los usuarios (o varias sincronizadas), y que éstos puedan entrar en cualquier máquina disponible del conjunto administrado con estos sistemas. Además estos sistemas incorporan conceptos adicionales de jerarquías, y/o dominios/zonas de máquinas y recursos, que permiten representar adecuadamente los recursos y su uso en organizaciones con diferentes estructuras de organización de su propio personal y sus secciones internas.

Podemos comprobar si estamos en un entorno de tipo NIS si en las líneas passwd y group del archivo de configuración /etc/nsswitch.conf aparece compat, si estamos trabajando con los ficheros locales, o bien nis o nisplus según el sistema con que estemos trabajando. En general, para el usuario simple no supo-

ne ninguna modificación, ya que la gestión de las máquinas le es transparente, y más si se combina con ficheros compartidos por NFS que permite disponer de su cuenta sin importar con qué máquina trabaje. La mayor parte de los comandos anteriores pueden seguir usándose sin problema bajo NIS o NIS+, son equivalentes a excepción del cambio de contraseña, que en lugar de passwd, se suele hacer con yppasswd (NIS) o nispasswd (NIS+); aunque suele ser habitual que el administrador los renombre (por un enlace) a passwd, con lo cual los usuarios no notarán la diferencia.

Veremos éste y otros modos de configuración en las unidades de administración de red.

6. Servidores de impresión

El sistema de impresión de GNU/Linux [Gt] [Smi02] está heredado de la variante BSD de UNIX; este sistema se denominaba LPD (*line printer daemon*). Éste un sistema de impresión muy potente, ya que integra capacidades para gestionar tanto impresoras locales como de red. Y ofrece dentro del mismo, tanto el cliente como el servidor de impresión.

LPD es un sistema bastante antiguo, ya que se remonta a los orígenes de la rama BSD de UNIX (mediados de los ochenta). Por lo tanto, a LPD le suele faltar soporte para los dispositivos modernos, ya que en origen el sistema no estuvo pensado para el tipo de impresoras actuales. El sistema LPD no estuvo pensado como un sistema basado en controladores de dispositivo, ya que normalmente se producían sólo impresoras serie o paralelas de escritura de caracteres texto.

Para la situación actual, el sistema LPD se combina con otro software común, como el sistema Ghostscript, que ofrece salida de tipo *postscript* para un rango muy amplio de impresoras para las que posee controladores. Además, se suele combinar con algún software de filtraje, que según el tipo de documento a imprimir, selecciona filtros adecuados. Así, normalmente el proceso que se sigue es (básicamente):

Nota

Los sistemas UNIX disponen, quizás, de los sistemas de impresión más potentes y complejos, que aportan una gran flexibilidad a los entornos de impresión.

1) El trabajo es iniciado por un comando del sistema LPD.

2) El sistema de filtro identifica qué tipo de trabajo (o fichero) es utilizado y convierte el trabajo a un fichero *postscript* de salida, que es el que se envía a la impresora. En GNU/Linux y UNIX, la mayoría de aplicaciones suponen que la salida será hacia una impresora *postscript*, y muchas de ellas generan salida *postscript* directamente, y por esta razón se necesita el siguiente paso.

3) Ghostscript se encarga de interpretar el fichero *postscript* recibido, y según el controlador de la impresora a la que ha sido enviado el trabajo, realiza la conversión al formato propio de la impresora; si es de tipo *postscript*, la impresión es directa, si no, habrá que realizar la traducción. El trabajo se manda a la cola de impresión.

Nota

Ghostscript: http://www.cs.wisc.edu/ghost/

Además del sistema de impresión LPD (con origen en los BSD UNIX), también existe el denominado sistema System V (de origen en la otra rama UNIX de System V). Normalmente, por compatibilidad, la mayor parte de UNIX integran actualmente ambos sistemas, de manera que o bien uno u otro es el principal, y el otro se simula sobre el principal. En el caso de GNU/Linux, pasa algo parecido, según la instalación que hagamos, podemos tener sólo los comandos LPD de sistema de impresión, pero también será habitual disponer de los

comandos System V. Una forma sencilla de identificar los dos sistemas (BSD o System V), es con el comando principal de impresión (el que envía los trabajos al sistema), en BSD es lpr, y en System V es lp.

Éste era el panorama inicial de los sistemas de impresión de GNU/Linux, pero en los últimos años han surgido más sistemas, que permiten una mayor flexibilidad y una mayor disposición de controladores para las impresoras. Los dos principales sistemas son CUPS, y en menor grado LPRng. Siendo, de hecho, últimamente CUPS el estándar de facto para GNU/Linux, aunque los otros sistemas han de ser soportados por compatibilidad con sistemas UNIX existentes.

Nota

LPRng: http://www.lprng.org
CUPS: http://www.cups.org

Los dos (tanto CUPS como LPRng) son una especie de sistemas de más alto nivel, pero que no se diferencian en mucho de cara al usuario respecto a los BSD y System V estándar, por ejemplo, se utilizan los mismos comandos clientes (o compatibles en opciones) para imprimir. Para el administrador sí que supondrá diferencias, ya que los sistemas de configuración son diferentes. En cierto modo podemos considerar a LPRng y CUPS como nuevas arquitecturas de sistemas de impresión, que son compatibles de cara al usuario con los comandos antiguos.

En las distribuciones GNU/Linux actuales podemos encontrarnos con los diferentes sistemas de impresión. Si la distribución es antigua, puede que lleve incorporado tan sólo el sistema BSD LPD; en las actuales: tanto Debian como Fedora/Red Hat utilizan CUPS. En algunas versiones de Red Hat existía una herramienta, Print switch, que permitía cambiar el sistema, conmutar de sistema de impresión, aunque últimamente solo está disponible CUPS. En Debian pueden instalarse ambos sistemas, pero son exclusivos, sólo uno de ellos puede gestionar la impresión.

En el caso de Fedora Core, el sistema de impresión por defecto es CUPS (desapareciendo LPRng en Fedora Core 4), y la herramienta Print switch ya no existe por no ser necesaria, se utiliza system-config-printer para la configuración de dispositivos. Debian por defecto utiliza BSD LPD, pero es común instalar CUPS (y es previsible que sea la opción por defecto en nuevas revisiones), y también puede utilizar LPRng. Además, cabe recordar que también teníamos la posibilidad (vista en la unidad de migración) de interaccionar con sistemas Windows mediante protocolos Samba, que permitían compartir las impresoras y el acceso a éstas.

Respecto a cada uno de los sistemas [Gt]:

- BSD LPD: es uno de los estándares de UNIX, y algunas aplicaciones asumen que tendrán los comandos y el sistema de impresión disponibles, por lo cual, tanto LPRng como CUPS emulan el funcionamiento y los comandos de BDS LPD. El sistema LPD es utilizable, pero no muy configurable, sobre todo en el control de acceso, por eso las distribuciones se han movido a los otros sistemas más modernos.

- LPRng: se diseñó básicamente para ser un reemplazo del BSD, por lo tanto, la mayor parte de la configuración es parecida y únicamente difiere en algunos ficheros de configuración.

- CUPS: se trata de una desviación mayor del BSD original, y la configuración es propia. Se proporciona información a las aplicaciones sobre las impresoras disponibles (también en LPRng). En CUPS tanto el cliente como el servidor tienen que disponer de software CUPS.

Los dos sistemas tienen emulación de los comandos de impresión de System V.

Para la impresión en GNU/Linux, hay que tener en cuenta varios aspectos:

- Sistema de impresión que se utiliza: BSD, LPRng o CUPS.

- Dispositivo de impresión (impresora): puede disponer de conexión local a una máquina o estar colocada en red. Las impresoras actuales pueden estar colocadas por conexiones locales a una máquina mediante interfaces serie, paralelo, USB, etc. O puestas simplemente en red, como una máquina más, o con protocolos especiales propietarios. Las conectadas a red, pueden normalmente actuar ellas mismas de servidor de impresión (por ejemplo, muchas láser HP son servidores BSD LPD), o bien pueden colgarse de una máquina que actúe de servidor de impresión para ellas.

- Protocolos de comunicación utilizados con la impresora, o el sistema de impresión: ya sea TCP/IP directo (por ejemplo, una HP con LPD), o bien otros de más alto nivel sobre TCP/IP, como IPP (CUPS), JetDirect (algunas impresoras HP), etc. Este parámetro es importante, ya que lo debemos conocer para instalar la impresora en un sistema.

- Sistema de filtros usados: cada sistema de impresión soporta uno o varios.

- Controladores de las impresoras: en GNU/Linux hay bastantes tipos diferentes, podemos mencionar, por ejemplo, controladores de CUPS, propios o de los fabricantes (por ejemplo HP y Epson los proporcionan); Gimp, el programa de retoque de imágenes también posee controladores optimizados para la impresión de imágenes; Foomatic es un sistema de gestión de controladores que funciona con la mayoría de sistemas (CUPS, LPD, LPRng, y otros); los controladores de Ghostscript, etc. En casi todas las impresoras tienen uno o más controladores de estos conjuntos.

Nota

Se puede encontrar información de las impresoras más adecuadas y de los controladores en: http://www.linuxprinting.org/foomatic.html

Respecto a la parte cliente del sistema, los comandos básicos son iguales para los diferentes sistemas, y éstos son los comandos del sistema BSD (cada sistema soporta emulación de estos comandos:

- lpr: envía un trabajo a la cola de la impresora por defecto (o a la que se selecciona), el *daemon* de impresión (lpd) se encarga de enviarlo a la cola co-

rrespondiente, y asigna un número de trabajo, que será usado con los otros comandos. Normalmente, la impresora por defecto estaría indicada por una variable de sistema PRINTER, o se utilizará la primera que exista definida, o en algunos sistemas se utiliza la cola lp (como nombre por defecto).

Ejemplo

Ejemplo de lpr:

```
lpr -Pepson datos.txt
```

Esta instrucción mandaría el fichero datos.txt a la cola de impresión asociada a una impresora que hemos definido como "epson".

- lpq: nos permite examinar los trabajos existentes en la cola.

Ejemplo

Ejemplo

```
# lpq -P epson
Rank   Owner      Job Files    Total          Size
1st    juan       15           datos.txt      74578 bytes
2nd    marta      16           fpppp.F        12394 bytes
```

Este comando nos muestra los trabajos en cola, con el orden y tamaños de éstos; los ficheros pueden aparecer con nombres diferentes, ya que depende de si los hemos enviado con lpr, o con otra aplicación que puede cambiarlos de nombre al enviarlos, o si han tenido que pasar por algún filtro al convertirlos.

- lprm*:* elimina trabajos de la cola, podemos especificar un número de trabajo, o un usuario para cancelar los trabajos.

Ejemplo

```
lprm -Pepson 15
```

Eliminar el trabajo con id 15 de la cola.

Respecto a la parte administrativa (en BSD), el comando principal sería lpc*;* este comando permite activar, desactivar colas, mover trabajos en el orden de las colas y activar o desactivar las impresoras (se pueden recibir trabajos en las colas pero no se envían a las impresoras).

Cabe mencionar asimismo que, para el caso de System V, los comandos de impresión suelen también estar disponibles, normalmente simulados sobre los de BSD. En el caso cliente, los comandos son: lp, lpstat, cancel y para temas de administración: lpadmin, accept, reject, lpmove, enable, disable, lpshut.

En las siguientes secciones veremos cómo hay que configurar un servidor de impresión para los tres sistemas principales. Estos servidores sirven tanto para la impresión local, como para atender las impresiones de clientes de red (si están habilitados).

6.1. BSD LPD

En el caso del servidor BSD LPD, hay dos ficheros principales para examinar, por una parte, la definición de las impresoras en /etc/printcap, y por otra, los permisos de acceso por red en /etc/hosts.lpd.

Respecto al tema de los permisos, por defecto BSD LPD sólo deja acceso local a la impresora, y por lo tanto, hay que habilitarlo expresamente en /etc/hosts.lpd.

Ejemplo

El fichero podría ser:

```
#fichero hosts.lpd
second
first.the.com
192.168.1.7
+@groupnis
-three.the.com
```

que indicaría que está permitida la impresión a una serie de máquinas, listadas bien por su nombre DNS o por la dirección IP. Se pueden añadir grupos de máquinas que pertenezcan a un servidor NIS (como en el ejemplo groupnis) o bien no permitir acceso a determinadas máquinas indicándolo con un guión "-".

Respecto a la configuración del servidor en /etc/printcap, se definen entradas, donde cada una representa una cola del sistema de impresión a la que pueden ir a parar los trabajos. La cola puede estar tanto asociada a un dispositivo local, como a un servidor remoto, ya sea éste una impresora u otro servidor.

En cada entrada pueden existir las opciones:

- lp =, nos indica a qué dispositivo está conectada la impresora, por ejemplo lp = /dev/lp0 indicaría el primer puerto paralelo. Si la impresora es de tipo LPD, por ejemplo una impresora de red que acepta el protocolo LPD (como una HP), entonces podemos dejar el campo vacío y rellenar los siguientes.

- rm =, dirección con nombre o IP de la máquina remota que dispone de la cola de impresión. Si se trata de una impresora de red, será la dirección de ésta.

- rp =, nombre de la cola remota, en la máquina indica antes con rm.

Veamos un ejemplo:

```
# Entrada de una impresora local
lp|epson|Epson C62:\
 :lp=/dev/lp1:sd=/var/spool/lpd/epson:\
 :sh:pw#80:pl#72:px#1440:mx#0:\
 :if = /etc/magicfilter/StylusColor@720dpi-filter:\filtro
 :af = /var/log/lp-acct:lf = /var/log/lp-errs:
```

```
# Entrada de impresora remota
hpremota|hpr|hp remota del departamento|:\
 :lp = :\
 :rm = servidor:rp = colahp:\
 :lf = /var/adm/lpd_rem_errs:\fichero de log.
 :sd = /var/spool/lpd/hpremota:spool local asociado
```

6.2. LPRng

En el caso del sistema LPRng, ya que éste se hizo para mantener la compatibilidad con BSD, y entre otros mejorar aspectos de acceso, el sistema es compatible a nivel de configuración de colas, y se lleva a cabo a través del mismo formato de fichero de /etc/printcap, con algunos añadidos propios.

Donde la configuración resulta diferente es en el tema del acceso, en este caso se realiza a través de un fichero /etc/lpd.perms en general para todo el sistema, y pueden existir también configuraciones individuales de cada cola, con el fichero lpd.perms, colocado en el directorio correspondiente a la cola, normalmente /var/spool/lpd/nombre-cola.

Estos ficheros lpd.perms tienen una capacidad superior de configurar el acceso, permitiendo los siguientes comandos básicos:

```
DEFAULT ACCEPT
DEFAULT REJECT
ACCEPT [ key = value[,value]* ]*
REJECT [ key = value[,value]* ]*
```

donde los dos primeros nos permiten establecer el valor por defecto, de aceptar todo, o rechazar todo, y los dos siguientes, aceptar o rechazar una configuración concreta especificada en la línea. Se pueden aceptar (o rechazar) peticiones de un *host*, usuario, o puertos IP específicos. Asimismo, se puede configurar qué tipo de servicio se proporcionará al elemento: X (puede conectarse), P (impresión de trabajos), Q (examinar cola con lpq), M (borrar trabajos de la cola, lprm), C (control de impresoras, comando lpc), entre otros, así en el fichero:

```
ACCEPT SERVICE = M HOST = first USER = jose
ACCEPT SERVICE = M SERVER REMOTEUSER = root
REJECT SERVICE = M
```

Se permite borrar trabajos de la cola, al usuario (jose) de la máquina (*first*), y al usuario *root*, del servidor donde esté alojado el servicio de impresión (*localhost*), además, se rechazan cualesquiera otras peticiones de borrar de la cola trabajos que no sean las peticiones ya establecidas.

Con esta configuración hay que tener especial cuidado, porque en algunas distribuciones, los servicios LPRng están por defecto abiertos. Puede limitarse la conexión por ejemplo con:

```
ACCEPT SERVICE = X SERVER
REJECT SERVICE = X NOT REMOTEIP = 100.200.0.0/255
```

Servicio de conexión sólo accesible a la máquina local del servidor, y rechazado si no se pertenece a nuestra subred (en este caso, suponemos que sea 100.200.0.x).

Para la administración de línea de comandos, se usan las mismas herramientas que el BSD, estándar. En el terreno de la administración gráfica del sistema, cabe destacar la herramienta *lprngtool* (no disponible en todas las versiones del sistema LPRng).

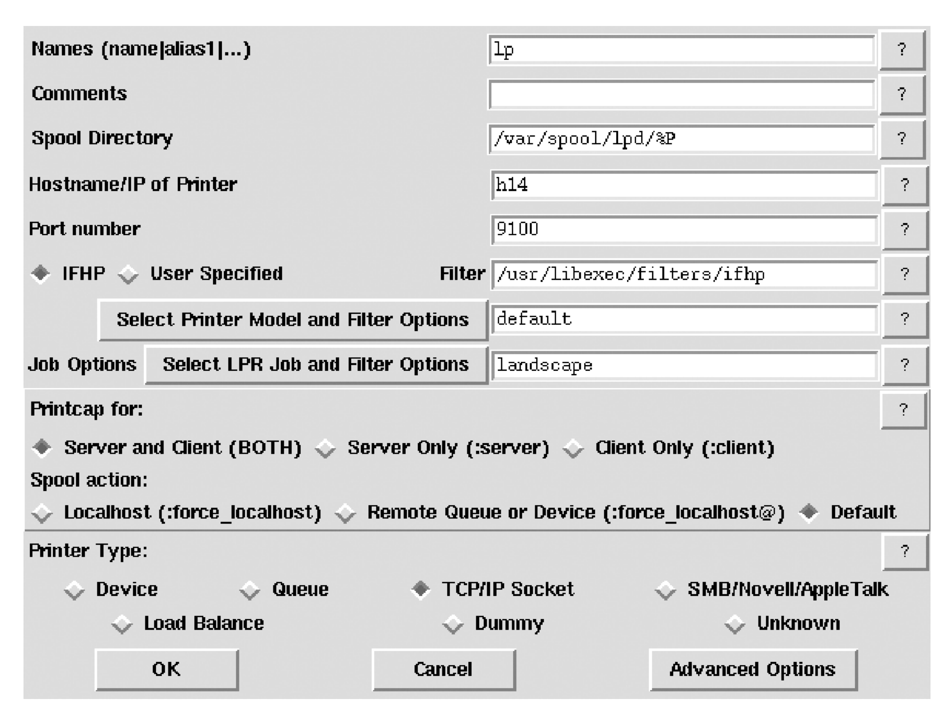


Figura 1. lprngtool, configuración de una impresora

Hay varios paquetes de software relacionados con LPRng, por ejemplo en una Debian encontramos:

```
lprng - lpr/lpd printer spooling system
lprng-doc - lpr/lpd printer spooling system (documentation)
lprngtool - GUI frontend to LPRng based /etc/printcap
printop - Graphical interface to the LPRng print system.
```

6.3. CUPS

CUPS es una nueva arquitectura para el sistema de impresión bastante diferente, tiene una capa de compatibilidad hacia BSD LPD, que le permite interaccionar con servidores de este tipo. Soporta también un nuevo protocolo de impresión

llamado IPP (basado en http), pero sólo disponible cuando cliente y servidor son de tipo CUPS. Además, utiliza un tipo de *drivers* denominados PPD que identifican las capacidades de la impresora, CUPS ya trae algunos de estos controladores, y algunos fabricantes también los ofrecen (caso HP y Epson).

CUPS tiene un sistema de administración completamente diferente, basado en diferentes ficheros: /etc/cups/cupsd.conf centraliza la configuración del sistema de impresión, y /etc/cups/printers.conf controla la definición de impresoras, y /etc/cups/classes.conf los grupos de éstas.

En /etc/cups/cupsd.conf, configuramos el sistema según una serie de secciones del archivo y las directivas de las diferentes acciones. El archivo es bastante grande, destacaremos algunas directivas importantes:

- Allow: nos permite especificar qué máquinas podrán acceder al servidor, ya sean grupos o máquinas individuales, o segmentos IP de red.

- AuthClass: permite indicar si se pedirá que se autentifiquen los usuarios clientes o no.

- BrowseXXX: hay una serie de directivas relacionadas con la posibilidad de examinar la red para encontrar impresoras servidas, esta posibilidad está activada por defecto (*browsing en on*), por lo tanto, normalmente encontraremos disponibles todas las impresoras disponibles en la red. Podemos desactivarla, para solamente observar las impresoras que hayamos definido. Otra opción importante es BrowseAllow, que dice a quién le damos la posibilidad de preguntar por nuestras impresoras; por defecto está habilitada, por lo que cualquiera puede ver nuestra impresora desde nuestra red.

Destacar que CUPS en principio está pensado para que tanto clientes, como el servidor funcionen bajo el mismo sistema, si los clientes utilizan LPD o LPRng, hay que instalar un *daemon* de compatibilidad llamado cups-lpd (normalmente en paquetes como cupsys-bsd). En este caso, CUPS acepta trabajos que provengan de un sistema LPD o LPRng, pero no controla los accesos (cupsd.conf sólo sirve para el propio sistema CUPS), por lo tanto, habrá que implementar alguna estrategia de control de acceso, tipo *firewall* por ejemplo (ver unidad de seguridad).

Para la administración desde línea de comandos, CUPS es un tanto peculiar, ya que acepta tanto comandos LPD como System V en los clientes, y la administración suele hacerse con el comando lpadmin de SystemV. En cuanto a herramientas gráficas, disponemos de gnome-cups-manager, gtklp o la interfaz por web que trae el mismo sistema CUPS, accesible en http://localhost:631.



Figura 2. Interfaz para la administración del sistema CUPS

Respecto a los paquetes software relacionados con CUPS, en una Debian encontramos (entre otros):

```
cupsys - Common UNIX Printing System(tm) - server
cupsys-bsd - Common UNIX Printing System(tm) - BSD commands
cupsys-client - Common UNIX Printing System(tm) - client
programs (SysV)
cupsys-driver-gimpprint - Gimp-Print printer drivers for CUPS
cupsys-pt - Tool for viewing/managing print jobs under CUPS
cupsomatic-ppd - linuxprinting.org printer support -
transition package
foomatic-db - linuxprinting.org printer support - database
foomatic-db-engine - linuxprinting.org printer support -
programs
foomatic-db-gimp-print - linuxprinting - db Gimp-Print
printer drivers
foomatic-db-hpijs - linuxprinting - db HPIJS printers
foomatic-filters - linuxprinting.org printer support -
filters
foomatic-filters-ppds - linuxprinting - prebuilt PPD files
foomatic-gui - GNOME interface for Foomatic printer filter
system
gimpprint-doc - Users' Guide for GIMP-Print and CUPS
gimpprint-locales - Locale data files for gimp-print
gnome-cups-manager - CUPS printer admin tool for GNOME
gtklp - Frontend for cups written in gtk
```

7. Discos y gestión filesystems

Respecto a las unidades de almacenamiento, como hemos ido examinando poseen una serie de dispositivos asociados, dependiendo del tipo de interfaz:

- IDE: dispositivos

  /dev/hda disco máster, primer conector IDE;

  /dev/hdb disco *slave* del primer conector,

  /dev/hdc máster segundo conector,

  /dev/hdd *slave* segundo conector.

- SCSI: dispositivos /dev/sda, /dev/sdb... siguiendo la numeración que tengan los periféricos en el Bus SCSI.

- Disquetes: dispositivos /dev/fdx, con x número de disquetera (comenzando en 0). Hay diferentes dispositivos dependiendo de la capacidad del disquete, por ejemplo, el disquete de 1.44 MB en la disquetera A sería /dev/fd0H1440.

Respecto a las particiones presentes, el número que sigue al dispositivo representa el índice de la partición dentro del disco, y es tratado como un dispositivo independiente: /dev/hda1 primera partición del primer disco IDE, o /dev/sdc2, segunda partición del tercer dispositivo SCSI. En el caso de los discos IDE, éstos permiten cuatro particiones denominadas primarias, y un mayor número en lógicas. Así, si /dev/hdan, *n* será inferior o igual a 4, se tratará de una partición primaria, si no, se tratará de una partición lógica con *n* superior o igual a 5.

Con los discos y los sistemas de ficheros (*filesystems*) asociados, los procesos básicos que podemos realizar los englobamos en:

- Creación de particiones, o modificación de éstas. Mediante comandos como fdisk o parecidos (cfdisk, sfdisk).

- Formateo de disquetes: en caso de disquetes, pueden utilizarse diferentes herramientas: fdformat (formateo de bajo nivel), superformat (formateo a diferentes capacidades en formato msdos), mformat (formateo específico creando *filesystem* msdos estándar).

- Creación de *filesystems* linux, en particiones, mediante el comando mkfs. Hay versiones específicas para crear filesystems diversos mkfs.ext2, mkfs.ext3, y

también *filesystems* no linux: mkfs.ntfs, mkfs.vfat, mkfs.msdos, mkfs.minix, u otros. Para CD-ROM como mkisofs para crear los iso9660 (con extensiones joliet o rock ridge), que puedan ser una imagen de lo que después se acabará grabando sobre un CD/DVD, y junto con comandos como cdrecord permitirá finalmente crear/grabar los CD/DVD. Otro caso particular es la orden mkswap, que permite crear áreas de swap en particiones, que después se pueden activar o desactivar con swapon y swapoff.

- Montaje de los filesystems: comandos *mount, umount*.

- Verificación de estado: la principal herramienta de verificación de filesystems Linux es el comando *fsck*. Este comando comprueba las diferentes áreas del sistema de ficheros para verificar la consistencia y comprobar posibles errores y, en los casos que sea posible, corregirlos. El propio sistema activa automáticamente el comando en el arranque cuando detecta situaciones donde se ha producido una parada incorrecta (un apagón eléctrico o accidental de la máquina), o bien pasado un cierto número de veces en que el sistema se ha arrancado; esta comprobación suele comportar cierto tiempo, normalmente algunos minutos (dependiendo del tamaño de datos). También existen versiones particulares para otros sistemas de ficheros: fsck.ext2, fsck.ext3, fsck.vfat, fsck.msdos, etc. El proceso del fsck normalmente se realiza con el dispositivo en modo de “sólo lectura” con particiones montadas; se recomienda desmontar las particiones para realizar el proceso si se detectan errores y hay que aplicar correcciones. En determinados casos, por ejemplo si el sistema por comprobar es la raíz / y se detecta algún error crítico, se nos pedirá que cambiemos de modo de ejecución del sistema (*runlevel*) hacia modo sólo *root*, y hagamos allí la verificación. En general, si hay que hacer la verificación, se recomienda hacer éstas en modo superusuario (podemos conmutar en modo *runlevel* con los comandos init o telinit).

- Procesos de *backup*: ya sean del disco, bloques de disco, particiones, *filesystems*, ficheros... Hay varias herramientas útiles para ello: tar nos permite copiar ficheros hacia un fichero o a unidades de cinta; cpio, de forma parecida, puede realizar backups de ficheros hacia un fichero; tanto cpio como tar mantienen información de permisos y propietarios de los ficheros; dd permite copias, ya sea de ficheros, dispositivos, particiones o discos a fichero; es un poco complejo y hay que conocer información de bajo nivel, tipo, tamaño, bloque o sector, y puede enviarse también a cintas.

- Utilidades diversas: algunos comandos individuales, algunos de ellos utilizados por los procesos anteriores para hacer tratamientos diversos: *badblocks* para encontrar bloques defectuosos en el dispositivo; dumpe2fs para obtener información sobre *filesystems* Linux; tune2fs permite hacer procesos de *tunning* de filesystems Linux de tipo ext2 o ext3 y ajustar diferentes parámetros de comportamiento.

A continuación destacamos dos temas relacionados con la concepción del espacio de almacenamiento, que son utilizados en varios ambientes para la creación base del espacio de almacenamiento. El uso de RAID software, y la creación de volúmenes dinámicos.

7.1. RAID software

Las configuraciones de discos mediante esquemas RAID es uno de los esquemas de almacenamiento de alta disponibilidad más usados actualmente, cuando disponemos de varios discos para implementar nuestros sistemas de ficheros.

El enfoque principal de las diferentes técnicas existentes se basa en la tolerancia a fallos que se proporciona desde un nivel de dispositivo, el conjunto de discos, a diferentes tipos posibles de fallos, tanto físicos como de sistema, para evitar las pérdidas de datos o los fallos de coherencia en el sistema. Así como en algunos esquemas que están diseñados para aumentar las prestaciones del sistema de discos, ampliando el ancho de banda de éstos disponible hacia el sistema y las aplicaciones.

Hoy en día podemos encontrar RAID en hardware principalmente en servidores empresariales (aun cuando comienzan a tener cierta presencia en equipos de escritorio), donde se encuentran disponibles diferentes soluciones hardware que cumplen estos requisitos. En particular para aplicaciones intensivas en disco, como *streaming* de audio y/o vídeo, o grandes bases de datos.

En general, este hardware se encuentra en forma de tarjetas (o integradas en la máquina) de tipo controladoras RAID de discos, que implementan la gestión de uno o más niveles (de la especificación RAID), sobre un conjunto de discos administrado por esta controladora.

En RAID se distinguen una serie de niveles (o configuraciones posibles) que pueden proporcionarse (cada fabricante de hardware, o el software concreto, puede soportar uno o varios de estos niveles). Cada nivel de RAID se aplica sobre un conjunto de discos, a veces denominado *array* RAID (o matriz de discos RAID), los cuales suelen ser discos iguales en tamaño (o iguales a tamaños de grupos). Por ejemplo, para realizar un caso de *array* podrían utilizarse 4 discos de 100GB, o en otro caso, por ejemplo, 2 grupos (a 100GB) de 2 discos, uno de 30GB y otro de 70GB. En algunos casos de controladores hardware no se permite que los discos (o en grupos) sean de diferentes tamaños, en otros pueden utilizarse, pero el *array* queda definido por el tamaño del disco (o grupo) más pequeño.

Describimos conceptos básicos de algunos niveles en la siguiente lista (téngase en cuenta que, en algunos casos, la terminología no es plenamente aceptada, y puede depender de cada fabricante):

- RAID 0: Se distribuyen los datos equitativamente entre uno o más discos sin información de paridad o redundancia, no se está ofreciendo tolerancia al fallo. Sólo se están repartiendo datos, si el disco falla físicamente, la in-

formación se pierde y debemos recuperarla desde copias de seguridad. Lo que sí que aumenta es el rendimiento, dependiendo de la implementación de RAID0, ya que las operaciones de lectura y escritura se dividirán entre los diferentes discos.

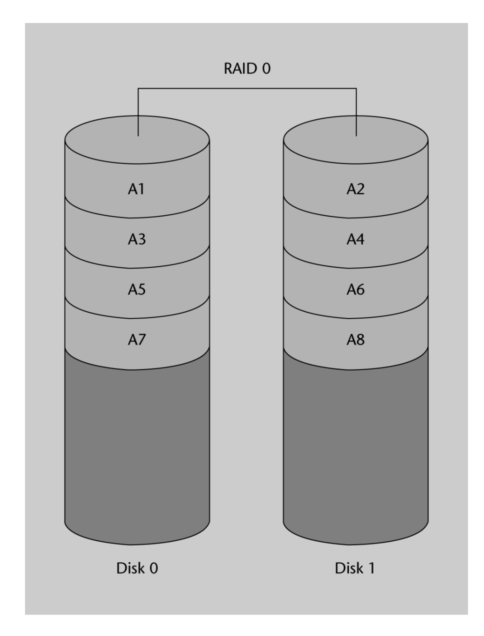


Figura 3

- RAID 1: Se crea una copia exacta (*mirror*) en un conjunto de dos o más discos (denominado array RAID). En este caso resulta útil para el rendimiento de lectura (que puede llegar a incrementarse de forma lineal con el número de discos), y en especial por disponer de tolerancia al fallo de uno de los discos, ya que (por ejemplo, con dos discos) se dispone de la misma información. RAID 1, suele ser adecuado para alta disponibilidad, como entornos de 24x7, donde debemos disponer críticamente de los recursos. Esta configuración nos permite también (si el hardware lo soporta) el intercambio en caliente de los discos. Si detectamos el fallo en uno de ellos, podemos sustituirlo sin apagar el sistema, por un disco nuevo.

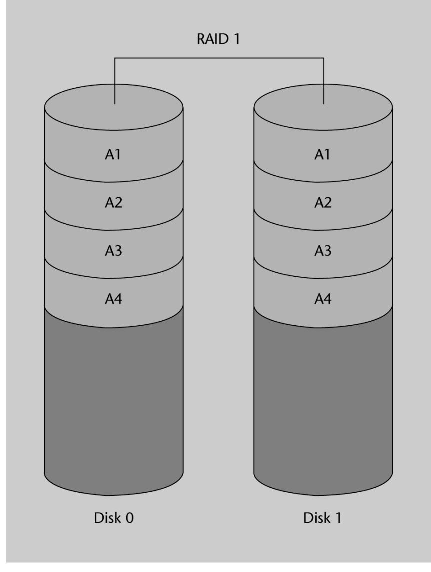


Figura 4

- RAID 2: En los anteriores se divide los datos en bloques a repartir, aquí, se divide en bits y se utilizan códigos de redundancia, para la corrección de datos. Prácticamente no se utiliza, a pesar de las altas prestaciones que alcanzaría, ya que necesita idealmente un número muy alto de discos, uno por bit de datos, y varios para el calculo de la redundancia (por ejemplo, en un sistema de 32 bits, llegaría a usar 39 discos).

- RAID 3: Utiliza división en bytes con un disco dedicado a la paridad de los bloques. Tampoco es muy utilizada, ya que según el tamaño de los datos y posiciones no permite accesos simultáneos. RAID 4 es semejante, aunque dividendo a nivel de bloques en lugar de bytes, permitiendo que sí que se puedan servir peticiones simultáneas cuando se solicita un único bloque.

- RAID 5: Se usa división a nivel de bloques, distribuyendo la paridad entre los discos. Tiene amplio uso, debido al esquema sencillo de paridad, y a que este calculo se implementa de forma sencilla por hardware, con buenas prestaciones.

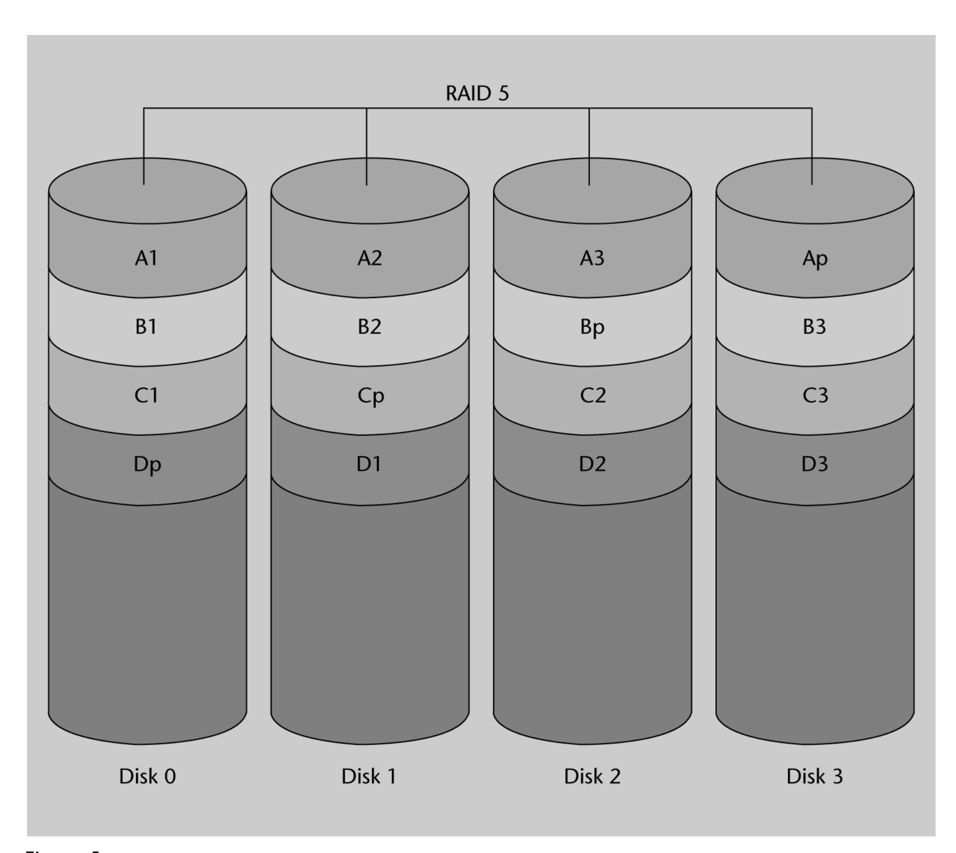


Figura 5

- RAID 0+1 (o 01): Un *mirror* de divisiones, se trata de un nivel de RAID anidado, se implementan, por ejemplo, dos grupos de RAID0, los cuales son usados en RAID 1 para crear *mirror* entre ellos. Una ventaja es que, en caso de fallo, puede reconstruirse el nivel de RAID0 usado gracias a la otra copia, pero si quieren añadirse discos hay que añadirlos a todos los grupos de RAID0 de igual forma.

- RAID 10 (1+0): división de *mirrors*, grupos de RAID 1 bajo RAID 0. Así, en cada grupo de RAID1 puede llegar a fallar un disco sin que se pierdan datos. Claro que esto obliga a reemplazarlos, ya que, si no, el disco que queda en el grupo se convierte en posible punto de fallo de todo el sistema. Es una configuración que suele usarse para base de datos de altas prestaciones (por la tolerancia a fallos, y la velocidad al no estar basada en cálculos de paridad).

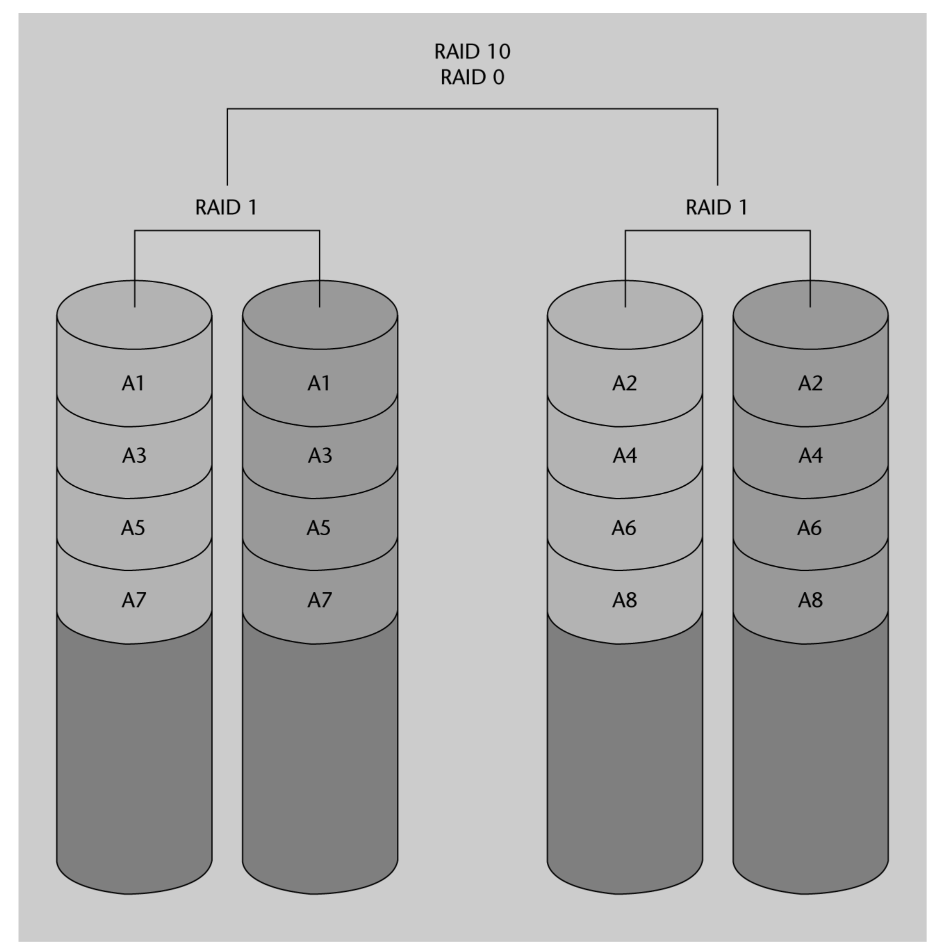


Figura 6

Algunas consideraciones a tener en cuenta sobre RAID en general:

- RAID mejora el *uptime* del sistema, ya que algunos de los niveles permiten que haya discos que fallan y el sistema siga siendo consistente, y dependiendo del hardware; incluso puede cambiarse el hardware problemático en caliente sin necesidad de parar el sistema, cuestión especialmente importante en sistema críticos.

- RAID puede mejorar el rendimiento de las aplicaciones, en especial en los sistemas con implementaciones de *mirror* es posible que la división de datos permite que las operaciones lineales de lectura se incrementen significativamente, debido a la posibilidad de que los discos ofrezcan simultáneamente partes de esta lectura, aumentando la tasa de transferencia de datos.

- RAID no protege los datos, evidentemente la destrucción por otros medios (virus, malfuncionamientos generales, o desastres naturales) no está protegida. Hemos de basarnos en esquemas de copias de seguridad.

- No se simplifica la recuperación de datos. Si un disco pertenece a un *array* RAID, tiene que intentar recuperarse en ese ambiente. Se necesita software específico o los controladores hardware para acceder a los datos.

- Por el contrario, no suele mejorar aplicaciones típicas de usuario, aunque sean de escritorio, debido a que estas aplicaciones tienen componentes altos de acceso aleatorio a datos, y a conjuntos de datos pequeños, por lo que puede que no se beneficien de lecturas lineales o de transferencias de datos sostenidas. En estos ambientes es posible que apenas se note mejoría de prestaciones.

- No se facilita el traslado de información, sin RAID es bastante fácil trasladar datos, simplemente moviendo el disco de un sistema a otro. En el caso de RAID es casi imposible (a no ser que dispongamos del mismo hardware), mover un *array* de discos a otro sistema.

En el caso de GNU/Linux, se da soporte al hardware RAID mediante diversos módulos de *kernel*, asociados a diferentes conjuntos de fabricantes o circuitos base, *chipsets*, de estas controladoras RAID. Permitiendo así al sistema abstraerse de los mecanismos hardware y hacerlos transparentes al sistema y al usuario final. En todo caso estos módulos de *kernel* nos permiten el acceso a los detalles de estas controladoras y a su configuración de parámetros de muy bajo nivel, que en algunos casos (especialmente en servidores que soportan carga elevada de E/S), pueden ser interesantes para procesos de *tunning* del sistema de discos que use el servidor, con la finalidad de maximizar las prestaciones del sistema.

La otra posibilidad que analizaremos aquí es la realización de estos procesos mediante componentes software, en concreto el componente software RAID de GNU/Linux.

GNU/Linux dispone en *kernel* del llamado Multiple Device (md), que podemos considerarlo como el soporte mediante *driver* del *kernel* para RAID. Mediante este *driver* podemos implementar niveles de RAID generalmente 0,1,4,5 y anidados (por ejemplo, RAID 10) sobre diferentes dispositivos de bloque como discos IDE o SCSI. También dispone del nivel *linear* como nivel donde se produce una combinación lineal de los discos disponibles (donde no importa que sean de diferentes tamaños), de manera que se escribe consecutivamente en los discos.

Para la utilización del RAID software en Linux, debemos disponer del soporte RAID en el *kernel*, y en su caso los modulos md activos (además de algunos *drivers* específicos según el caso (ver *drivers* disponibles asociados a RAID, por ejemplo en Debian con modconf). El método preferido para la implementación de *arrays* de discos RAID, mediante el software RAID ofrecido por Linux, es mediante (o bien durante la instalación) o bien mediante la utilidad mdadm. Esta utilidad nos permite crear los *arrays* y gestionarlos.

Veamos algunos ejemplos (supongamos unos discos SCSI /dev/sda, /dev/sdb... en los cuales disponemos de varias particiones disponibles para implementar RAID):

Creación de un *array linear:*

```
# mdadm –create –verbose /dev/md0 –level=linear –raid-devices=2
/dev/sda1 /dev/sdb1
```

donde se crea un *array* linear a partir de las particiones primeras de /dev/sda y /dev/sdb, creando el nuevo dispositivo /dev/md0, que ya puede ser usado como nuevo disco (suponiendo que exista el punto de montaje /media/discoRAID):

```
# mkfs.ext2fs /dev/md0
# mount /dev/md0 /media/discoRAID
```

Para un RAID0 o RAID1 podemos cambiar simplemente el nivel (-level) a raid0 o raid1. Con mdadm –detail /dev/md0 podremos comprobar los parámetros del nuevo *array* creado.

También podemos consultar la entrada mdstat en /proc para determinar los *arrays* activos, así como sus parámetros. En especial con los casos con *mirror* (por ejemplo en los niveles 1, 5...) podremos observar en su creación la reconstrucción inicial de las copias, en /proc/mdstat indicará el nivel de reconstrucción (y el tiempo aproximado de finalización).

mdadm dispone de muchas opciones que nos permiten examinar y gestionar los diferentes *arrays* RAID software creados (podemos ver una descripción y ejemplos en man mdadm).

Otra consideración importante son las optimizaciones a que se pueden someter los *arrays* RAID para mejorar su rendimiento, tanto por monitorizar su comportamiento por optimizar parámetros del sistema de ficheros, como para realizar un uso más efectivo de los niveles RAID y sus características.

Nota

La optimización de los arrays RAID, puede ser una fuente importante de sintonización del sistema, se recomienda examinar algunas cuestiones en:

Software-RAID-Howto, o en la propia página man de mdadm.

7.2. Volúmenes Lógicos (LVM)

Por otra parte surge la necesidad de abstraerse del sistema físico de discos, y su configuración, y numero de dispositivos, para que el sistema (operativo) se encargue de este trabajo, y no nos tengamos que preocupar de estos parámetros directamente. En este sentido, puede verse al sistema de volúmenes lógicos como una capa de virtualización del almacenamiento permitiendo una visión más simple que facilite la utilización fluida y sencilla.

En el kernel Linux se dispone de LVM (*logical volume manager*), que se basá a partir de ideas desarrolladas de gestores de volúmenes de almacenamiento usados en

HP-UX (una versión UNIX propietaria de HP). Actualmente existen dos versiones, siendo la LVM2 la más utilizada por una serie de prestaciones añadidas.

La arquitectura de una LVM consiste típicamente en los componentes (principales):

- **Volúmenes físicos** (PV): Son los discos duros, o particiones de éstos, o cualquier otro elemento que aparezca como un disco duro de cara al sistema (por ejemplo un RAID software o hardware).

- **Volúmenes lógicos** (LV): Es el equivalente a la partición del disco físico. Esta LV es visible en el sistema como un dispositivo de bloques (absolutamente equivalente a una partición física), y puede contener un sistema de ficheros (por ejemplo el /home de los usuarios). Normalmente los volúmenes tienen más sentido para los administradores, ya que pueden usarse nombres para identificárcarlos (así, podemos utilizar un dispositivo lógico, llamado stock, o marketing en lugar de hda6 o sdc3).

- **Grupos de volúmenes** (VG): Es el elemento de la capa superior. La unidad administrativa que engloba nuestros recursos, ya sean volúmenes lógicos (LV) o físicos (PV). En esta unidad se guardan los datos de los PV disponibles, y cómo se forman las LV a partir de los PV. Evidentemente, para poder utilizar un grupo VG, hemos de disponer de soportes físicos PV, que se organicen en diferentes unidades lógicas LV.

Por ejemplo, en la siguiente figura, observamos un grupo de volúmenes, donde disponemos de 7 PV (en forma de particiones de discos, que se han agrupado para formar dos volúmenes lógicos (que se han acabado utilizando para formar los sistemas de ficheros de /usr y /home) :

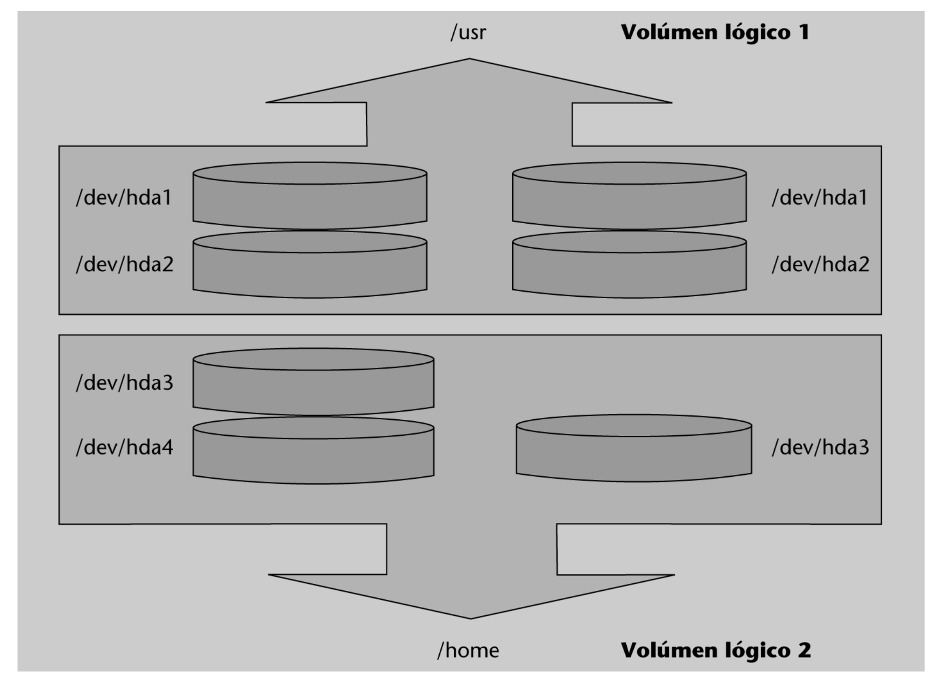


Figura 7. Esquema de un ejemplo de LVM

Con el uso de los volúmenes lógicos, permitimos un tratamiento más flexible del espacio en el sistema de almacenamiento (que podría tener un gran número de discos y particiones diferentes), según las necesidades que nos aparezcan, y poder gestionar el espacio, tanto por identificadores más adecuados como por operaciones que nos permitan adecuar las necesidades al espacio disponible en cada momento.

Los sistemas de gestión de volúmenes nos permiten:

- Redimensionar grupos y volúmenes lógicos, aprovechando nuevos PV, o extrayendo algunos de los disponibles inicialmente.

- Instantáneas del sistema de archivos (lectura en LVM1, y lectura y/o escritura en LVM2). Esto permite crear un nuevo dispositivo que sea una instantánea en el tiempo de la situación de una LV. Asimismo, permite crear la instantánea, montarla, probar diversas operaciones, o configuración nueva de software, u otros elementos, y si no funciona como esperábamos, devolver el volumen original a su estado antes de las pruebas.

- RAID0 de volúmenes lógicos.

En LVM no se implementan configuraciones de RAID de tipos 1 o 5, si son necesarias (o sea redundancia y tolerancia a fallo), entonces o bien se utiliza software de RAID o controladora hardware RAID que lo implemente, y colocamos LVM como capa superior.

Hagamos un breve ejemplo de creación típica (en muchos casos, el instalador de la distribución realiza un proceso parecido, si permitimos un LVM como sistema inicial de almacenamiento). Básicamente, se realiza: 1) la creación de los volúmenes físicos (PV). 2) la creación del grupo lógico (VG) y 3) la creación del volumen lógico, y finalmente la utilización para creación de un sistema de ficheros, y su posterior montaje:

1) ejemplo: disponemos de tres particiones de diferentes discos, creando tres PV e inicializando el contenido:

```
# dd if=/dev/zero of=/dev/hda1 bs=1k count=1
# dd if=/dev/zero of=/dev/hda2 bs=1k count=1
# dd if=/dev/zero of=/dev/hdb1 bs=1k count=1
# pvcreate /dev/hda1
Physical volume "/dev/sda1" successfully created
# pvcreate /dev/hda2
Physical volume "/dev/hda2" successfully created
# pvcreate /dev/hdb1
Physical volume "/dev/hdb1" successfully created
```

2) colocación en un VG creada de los diferentes PV:

```
# vgcreate grupo_discos /dev/hda1 /dev/hda2 /dev/hdb1
Volume group "grupo_discos" successfully created
```

3) creamos la LV (en este caso de tamaño de 1GB) a partir de los elementos que tenemos en el grupo VG (*-n* indica el nombre del volumen):

```
# lvcreate -L1G -n volumen_logico grupo_discos
lvcreate -- doing automatic backup of "grupo_discos"
lvcreate -- logical volume "/dev/grupo_discos/ volumen_logico"
successfully created
```

Y finalmente, creamos un sistema de ficheros (un *reiser* en este caso):

```
# mkfs.reiserfs /dev/grupo_discos/volumen_logico
```

El cual, por ejemplo, podríamos colocar de espacio de *backup*

```
mkdir /mnt/backup
mount -t reiserfs /dev/grupo_discos/volumen_logico /mnt/backup
```

Disponiendo finalmente del dispositivo como un volumen lógico que implementa un sistema de ficheros del nuestra máquina.

8. Software: actualización

Para la administración de instalación o actualización de software en nuestro sistema, vamos a depender en primera instancia del tipo de paquetes software que utilice nuestro sistema:

- RPM: paquetes que utiliza la distribución Fedora/Red Hat (y derivadas). Se suelen manejar a través del comando *rpm*. Contienen información de dependencias del software con otros. A alto nivel mediante *Yum* (o *up2date* en algunas distribuciones derivadas de Red Hat).

- DEB: paquetes de Debian, se suelen manejar con un conjunto de herramientas que trabajan a diferentes niveles con paquetes individuales o grupos. Entre éstas, cabe mencionar: dselect, tasksel, dpkg, y apt-get.

- Tar, o bien los tgz (también tar.gz): son puramente paquetes de ficheros unidos y comprimidos mediante comandos estándar como tar, y gzip (se usan éstos para la descompresión). Estos paquetes no contienen información de dependencias y normalmente pueden instalarse en diferentes lugares, si no es que llevan información de ruta (*path*) absoluta.

Para manejar estos paquetes, existen varias herramientas gráficas, como RPM: Kpackage; DEB: Synaptic, Gnome-apt; Tgz: Kpackage, o desde el propio gestor de ficheros gráficos (en Gnome o KDE). También suelen existir utilidades de conversiones de paquetes. Por ejemplo, en Debian tenemos el comando alien, que permite convertir paquetes RPM a DEB. Aunque hay que tomar las debidas precauciones, para que el paquete, por tener una distribución destino diferente, no modifique algún comportamiento o fichero de sistema no esperado.

Dependiendo del uso de los tipos de paquetes o herramientas: la actualización o instalación de software de nuestro sistema se podrá producir de diferentes maneras:

1) Desde los propios CD de instalación del sistema, normalmente, todas las distribuciones buscan el software en sus CD. Pero hay que tener en cuenta que este software no sea antiguo, y no incluya, por esta razón, algunos parches como actualizaciones, o nuevas versiones con más prestaciones; con lo cual, si se instala a partir de CD, es bastante común verificar después que no exista alguna versión más reciente.

2) Mediante servicios de actualización o búsqueda de software, ya sea de forma gratuita, como el caso de la herramienta apt-get de Debian, o yum en Fe-

dora, o servicios de suscripción (de pago o con facilidades básicas) como el Red Hat Network de las versiones Red Hat comerciales.

3) Por repositorios de software que ofrecen paquetes de software preconstruidos para una distribución determinada.

4) Por el propio creador o distribuidor del software, que ofrece una serie de paquetes de instalación de su software. Podemos no encontrar el tipo de paquetes necesario para nuestra distribución.

5) Software sin empaquetamiento o con un empaquetamiento de sólo compresión sin ningún tipo de dependencias.

6) Sólo código fuente, en forma de paquete o bien fichero comprimido.

9. Trabajos no interactivos

En las tareas de administración, suele ser necesaria la ejecución a intervalos de ciertas tareas, ya sea por programar las tareas para realizarlas en horarios de menor uso de la máquina, o bien por la propia naturaleza periódica de las tareas que se quieran desarrollar.

Para realizar este tipo de trabajos "fuera de horas", como servicios periódicos o programados, hay diversos sistemas que nos permiten construir un tipo de agenda de tareas (planificación de ejecución de tareas):

- nohup es quizás el caso más simple utilizado por los usuarios, les permite la ejecución de una tarea no interactiva una vez hayan salido de su cuenta. Normalmente, al salir de la cuenta, el usuario pierde sus procesos; nohup permite dejarlos en ejecución, a pesar de que el usuario se desconecte.

- at nos permite lanzar una acción para más tarde, programando un determinado instante en el que va a iniciarse, especificándose la hora (hh:mm) y fecha, o bien si se hará hoy (*today*) o mañana (*tomorrow*). Ejemplos:

  ```
  at 10pm tarea
  ```

  realizar la tarea a las diez de la noche.

  ```
  at 2am tomorrow tarea
  ```

  realizar la tarea a las dos de la madrugada.

- cron*:* permite establecer una lista de trabajos por realizar con su programación; esta configuración se guarda en /etc/crontab; concretamente, en cada entrada de este fichero tenemos: minutos y hora en que se va a efectuar la tarea, qué día del mes, qué mes, qué día de la semana, junto con qué (ya sea una tarea, o bien un directorio donde estarán las tareas a ejecutar). Por ejemplo, de forma estándar el contenido es parecido a:

```
25 6 * * * root test -e /usr/sbin/anacron || run-parts --report /etc/cron.daily
47 6 * * 7 root test -e /usr/sbin/anacron || run-parts --report /etc/cron.weekly
52 6 1 * * root test -e /usr/sbin/anacron || run-parts --report /etc/cron.monthl
```

donde se está programando que una serie de tareas van a hacerse: cada día ("*" indica 'cualquiera'), semanalmente (el 7.º día de la semana), o mensualmente (el 1.º de cada mes). Normalmente, los trabajos serían ejecutados por el comando crontab, pero el sistema cron supone que la máquina está siempre en-

cendida, si no es así, es mejor utilizar anacron, que verifica si la acción no se realizó cuando habría debido hacerlo, y la ejecuta. En cada línea del anterior fichero se verifica que exista el comando anacron y se ejecutan los *scripts* asociados a cada acción, que en este caso están guardados en unos directorios asignados para ello, los cron.

También pueden existir unos ficheros cron.allow, cron.deny, para limitar quién puede colocar (o no) trabajos en cron. Mediante el comando crontab, un usuario puede definir trabajos en el mismo formato que hemos visto antes, que habitualmente se guardarán en /var/spool/ cron/crontabs. En algunos casos existe también un directorio /etc/cron.d donde se pueden colocar trabajos y que es tratado como si fueran una extensión del fichero /etc/crontab.

10. Taller: prácticas combinadas de los diferentes apartados

Comenzaremos por examinar el estado general de nuestro sistema. Vamos a hacer los diferentes pasos en un sistema Debian. Se trata de un sistema Debian *unstable* (la versión inestable, pero más actualizada); sin embargo, los procedimientos son en su mayor parte trasladables a otras distribuciones como Fedora/Red Hat (mencionaremos algunos de los cambios más importantes). El hardware consiste en un Pentium 4 a 2.66 Ghz con 768 MB y varios discos, DVD y grabador de CD, además de otros periféricos, pero ya iremos obteniendo esta información paso a paso.

Veamos primero cómo ha arrancado nuestro sistema la última vez:

```
# uptime
17:38:22 up 2:46, 5 users, load average: 0.05, 0.03, 0.04
```

Este comando nos da el tiempo que lleva el sistema "levantado" desde que se arrancó la última vez, 2 horas 46 minutos, en nuestro caso tenemos cinco usuarios. Éstos no tienen por qué ser cinco usuarios diferentes, sino que normalmente serán las sesiones de usuario abiertas (por ejemplo, mediante un terminal). El comando who permite listar estos usuarios. El load average es la carga media del sistema en los últimos 1, 5 y 15 minutos.

Veamos el log del arranque del sistema (comando dmesg), las líneas que se iban generando en la carga del sistema (se han suprimido diferentes líneas por claridad):

```
Linux version 2.6.20-1-686 (Debian 2.6.20-2) (waldi@debian.org) (gcc
version 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)) #1 SMP Sun Apr
15 21:03:57 UTC 2007
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
 BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000ce000 - 00000000000d0000 (reserved)
 BIOS-e820: 00000000000dc000 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 000000002f6e0000 (usable)
 BIOS-e820: 000000002f6e0000 - 000000002f6f0000 (ACPI data)
 BIOS-e820: 000000002f6f0000 - 000000002f700000 (ACPI NVS)
 BIOS-e820: 000000002f700000 - 000000002f780000 (usable)
 BIOS-e820: 000000002f780000 - 0000000030000000 (reserved)
 BIOS-e820: 00000000ff800000 - 00000000ffc00000 (reserved)
 BIOS-e820: 00000000fffffc00 - 0000000100000000 (reserved)
0MB HIGHMEM available.
759MB LOWMEM available.
```

Estas primeras líneas ya nos indican varios datos interesantes: la versión del kernel Linux es la 2.6.20-1-686, una versión 2.6 revisión 20 a revisión 1 de De-

bian, y para máquinas 686 (arquitectura Intel 32bits). Indica también que estamos arrancando un sistema Debian, con este *kernel* que fue compilado con un compilador GNU gcc versión 4.1.2 y la fecha. A continuación, existe un mapa de zonas de memoria usadas (reservadas) por la BIOS, y a continuación el total de memoria detectada en la máquina: 759 MB, a las que habría que sumar el primer 1 MB, total de 760 MB.

```
Kernel command line: BOOT_IMAGE=LinuxNEW ro root=302 lang=es acpi=force
Initializing CPU#0
Console: colour dummy device 80x25
Memory: 766132k/777728k available (1641k kernel code, 10968k reserved, 619k data,
208k init, 0k highmem)
Calibrating delay using timer specific routine.. 5320.63 BogoMIPS (lpj=10641275)
```

Aquí nos refiere cómo ha sido el arranque de la máquina, qué línea de comandos se le ha pasado al *kernel* (pueden pasarse diferentes opciones, por ejemplo desde el lilo o grub). Y estamos arrancando en modo consola de 80 x 25 caracteres (esto se puede cambiar). Los BogoMIPS son una medida interna del *kernel* de la velocidad de la CPU. Hay arquitecturas donde es difícil detectar con cuántos MHz funciona la CPU, y por eso se utiliza esta medida de velocidad. Después nos da más datos de la memoria principal, y para qué se está usando en este momento del arranque.

```
CPU: Trace cache: 12K uops, L1 D cache: 8K
CPU: L2 cache: 512K
CPU: Hyper-Threading is disabled
Intel machine check architecture supported.
Intel machine check reporting enabled on CPU#0.
CPU0: Intel P4/Xeon Extended MCE MSRs (12) available
CPU0: Intel(R) Pentium(R) 4 CPU 2.66GHz stepping 09
```

Además, nos proporciona datos varios de la CPU, el tamaño de las caché de primer nivel, caché interna CPU, L1 dividida en una TraceCache del Pentium4 (o *instruction cache*), y la caché de datos, y caché unificada de segundo nivel (L2), tipo de CPU, velocidad de ésta y del bus del sistema.

```
PCI: PCI BIOS revision 2.10 entry at 0xfd994, last bus=3
Setting up standard PCI resources
...
NET: Registered protocol
IP route cache hash table entries: 32768 (order: 5, 131072 bytes)
TCP: Hash tables configured (established 131072 bind 65536)
checking if image is initramfs... it is
Freeing initrd memory: 1270k freed
fb0: VESA VGA frame buffer device
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing enabled
serial8250: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
00:09: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
RAMDISK driver initialized: 16 RAM disks of 8192K size 1024 blocksize
PNP: PS/2 Controller [PNP0303:KBC0,PNP0f13:MSE0] at 0x60,0x64 irq 1,12
i8042.c: Detected active multiplexing controller, rev 1.1.
serio: i8042 KBD port at 0x60,0x64 irq 1
serio: i8042 AUX0 port at 0x60,0x64 irq 12
serio: i8042 AUX1 port at 0x60,0x64 irq 12
serio: i8042 AUX2 port at 0x60,0x64 irq 12
serio: i8042 AUX3 port at 0x60,0x64 irq 12
mice: PS/2 mouse device common for all mice
```

Continúan las inicializaciones del *kernel* y dispositivos, menciona la inicialización de protocolos de red. Los terminales, los puertos serie ttyS0 (sería el com1), ttyS01 (el com2). Da información de los discos RAM que usamos, y detección de dispositivos PS2, teclado y ratón.

```
ICH4: IDE controller at PCI slot 0000:00:1f.1

 ide0: BM-DMA at 0x1860-0x1867, BIOS settings: hda:DMA, hdb:pio
 ide1: BM-DMA at 0x1868-0x186f, BIOS settings: hdc:DMA, hdd:pio
Probing IDE interface ide0...
hda: FUJITSU MHT2030AT, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
Probing IDE interface ide1...
hdc: SAMSUNG CDRW/DVD SN-324F, ATAPI CD/DVD-ROM drive
ide1 at 0x170-0x177,0x376 on irq 15
SCSI subsystem initialized
libata version 2.00 loaded.
hda: max request size: 128KiB
hda: 58605120 sectors (30005 MB) w/2048KiB Cache, CHS=58140/16/63<6>hda:
hw_config=600b
, UDMA(100)
hda: cache flushes supported
 hda: hda1 hda2 hda3
kjournald starting. Commit interval 5 seconds
EXT3-fs: mounted filesystem with ordered data mode.
hdc: ATAPI 24X DVD-ROM CD-R/RW drive, 2048kB Cache, UDMA(33)
Uniform CD-ROM driver Revision: 3.20
Addinf 618492 swap on /dev/hda3.
```

Detección de dispositivos IDE, detecta el chip IDE en el bus PCI, e informa de que está controlando dos dispositivos: hda, y hdc, que son respectivamente: un disco duro (Fujitsu), un segundo disco duro, un DVD Samsung, y una grabadora de CD (ya que en este caso se trata de una unidad combo). Indica particiones activas. Más adelante, detecta el sistema principal de ficheros de Linux, un ext3 con *journal*, que activa y añade el espacio de swap disponible en una partición.

```
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
input: PC Speaker as /class/input/input1
USB Universal Host Controller Interface driver v3.0
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 2 ports detected
uhci_hcd 0000:00:1d.1: UHCI Host Controller
uhci_hcd 0000:00:1d.1: new USB bus registered, assigned bus number 2
uhci_hcd 0000:00:1d.1: irq 11, io base 0x00001820
usb usb2: configuration #1 chosen from 1 choice
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 2 ports detected
hub 4-0:1.0: USB hub found
hub 4-0:1.0: 6 ports detected
```

Más detección de dispositivos, USB en este caso (y los módulos que corresponden), ha detectado dos dispositivos hub (con un total de 8 USB ports).

```
parport: PnPBIOS parport detected.
parport0: PC-style at 0x378 (0x778), irq 7, dma 1
[PCSPP,TRISTATE,COMPAT,EPP,ECP,DMA]
input: ImPS/2 Logitech Wheel Mouse as /class/input/input2
ieee1394: Initialized config rom entry ‘ip1394'
```

```
eepro100.c:v1.09j-t 9/29/99 Donald Becker
Synaptics Touchpad, model: 1, fw: 5.9, id: 0x2e6eb1, caps:
0x944713/0xc0000
input: SynPS/2 Synaptics TouchPad as /class/input/input3

agpgart: Detected an Intel 845G Chipset
agpgart: Detected 8060K stolen Memory
agpgart: AGP aperture is 128M
eth0: OEM i82557/i82558 10/100 Ethernet, 00:00:F0:84:D3:A9, IRQ 11.
 Board assembly 000000-000, Physical connectors present: RJ45
e100: Intel(R) PRO/100 Network Driver, 3.5.17-k2-NAPI
usbcore: registered new interface driver usbkbd
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.

lp0: using parport0 (interrupt-driven).
ppdev: user-space parallel port driver
```

Y la detección final del resto de dispositivos: Puerto paralelo, modelo de ratón, puerto *firewire* (ieee1394) tarjeta de red (Intel), un panel táctil, la tarjeta de vídeo AGP (i845). Más datos de la tarjeta de red, una intel pro 100, registro de usb como mass storage (indica un dispositivo de almacenamiento por usb, como un disco externo), y detección del puerto paralelo.

Toda esta información que hemos visto mediante el comando dmesg, también la podemos encontrar volcada en el log principal del sistema /var/log/messages. En este log, entre otros, encontraremos los mensajes del *kernel* y de los *daemons*, y errores de red o dispositivos, los cuales comunican sus mensajes a un *daemon* especial llamado syslogd, que es el encargado de escribir los mensajes en este fichero. Si hemos arrancado la máquina recientemente, observaremos que las últimas líneas contienen exactamente la misma información que el comando dmesg,

por ejemplo, si nos quedamos con la parte final del fichero (suele ser muy grande):

```
# tail 200 /var/log/messages
```

Observamos las líneas de antes, y también algunas informaciones más, como por ejemplo:

```
shutdown[13325]: shutting down for system reboot
kernel: usb 4-1: USB disconnect, address 3
kernel: nfsd: last server has exited
kernel: nfsd: unexporting all filesystems
kernel: Kernel logging (proc) stopped.
kernel: Kernel log daemon terminating.

exiting on signal 15
syslogd 1.4.1#20: restart.

kernel: klogd 1.4.1#20, log source = /proc/kmsg started.
Linux version 2.6.20-1-686 (Debian 2.6.20-2) (waldi@debian.org) (gcc version 4.1.2
20061115 (prerelease) (Debian 4.1.1-21)) #1 SMP Sun Apr 15 21:03:57 UTC 2007
kernel: BIOS-provided physical RAM map:
```

La primera parte corresponde a la parada anterior del sistema, nos informa de que el *kernel* ha dejado de colocar información en /proc, se está parando el sis-

tema... Al principio del arranque nuevo, se activa el *daemon* Syslogd que genera el log, y comienza la carga del sistema, que nos dice que el *kernel* comenzará a escribir información en su sistema, /proc; vemos las primeras líneas de dmesg de mención de la versión que se está cargando de *kernel*, y luego encontraremos lo que hemos visto con dmesg.

En este punto, otro comando útil para saber cómo se ha producido la carga es ismod, que nos permitirá saber qué módulos se han cargado en el *kernel* (versión resumida):

```
# lsmod
Module Size Used by
nfs 219468      0
nfsd 202192     17
exportfs 5632   1 nfsd
lockd 58216     3 nfs,nfsd
nfs_acl 3616    2 nfs,nfsd
sunrpc 148380   13 nfs,nfsd,lockd,nfs_acl
ppdev 8740      0
lp 11044              0
button 7856     0
ac 5220               0
battery 9924    0
md_mod 71860    1
dm_snapshot 16580     0
dm_mirror 20340       0
dm_mod 52812    2 dm_snapshot,dm_mirror
i810fb 30268    0
vgastate 8512   1 i810fb
eeprom 7184     0
thermal 13928   0
processor 30536 1 thermal
fan 4772              0
udf 75876             0
ntfs 205364     0
usb_storage 75552     0
hid 22784             0
usbkbd 6752     0
eth1394 18468   0
e100 32648      0
eepro100 30096  0
ohci1394 32656  0
ieee1394 89208  2 eth1394,ohci1394
snd_intel8x0 31420    1
snd_ac97_codec 89412  1 snd_intel8x0
ac97_bus 2432   1 snd_ac97_codec
parport_pc 32772      1
snd 48196       6 snd_intel8x0,snd_ac97_codec,snd_pcm,snd_timer
ehci_hcd 29132   0
ide_cd 36672    0
cdrom 32960     1 ide_cd
soundcore 7616  1 snd
psmouse 35208   0
uhci_hcd 22160  0
parport 33672   3 ppdev,lp,parport_pc
intelfb 34596         0
serio_raw 6724  0
pcspkr 3264     0
pci_hotplug 29312     1 shpchp
usbcore 122312  6 dvb_usb,usb_storage,usbkbd,ehci_hcd,uhci_hcd
intel_agp 22748 1
agpgart 30504   5 i810fb,drm,intelfb,intel_agp
ext3 121032     1
jbd 55368             1 ext3
ide_disk 15744  3
ata_generic 7876      0
```

```
ata_piix 15044      0
libata 100052       2 ata_generic,ata_piix
scsi_mod 133100            2 usb_storage,libata
generic 4932        0 [permanent]
piix 9540                  0 [permanent]
ide_core 114728            5 usb_storage,ide_cd,ide_disk,generic,piix
```

Vemos que básicamente tenemos los controladores para el hardware que hemos detectado, y otros relacionados o necesarios por dependencias.

Ya tenemos, pues, una idea de cómo se ha cargado el *kernel* y sus módulos. En este proceso puede que ya hubiésemos observado algún error, si hay hardware mal configurado o módulos del *kernel* mal compilados (no eran para la versión del *kernel* adecuada), inexistentes, etc.

El paso siguiente será la observación de los procesos en el sistema, con el comando ps (*process status*), por ejemplo (sólo se han sacado los procesos de sistema, no los de los usuarios):

```
# ps -ef
UID PID PPID C STIME TTY TIME CMD
```

Información de los procesos, UID usuario que ha lanzado el proceso (o con qué identificador se ha lanzado), PID, código del proceso asignado por el sistema, son consecutivos a medida que se lanzan los procesos, el primero siempre es el 0, que corresponde al proceso de init. PPID es el id del proceso padre del actual. STIME, tiempo en que fue arrancado el proceso, TTY, terminal asignado al proceso (si tiene alguno), CMD, línea de comando con que fue lanzado.

```
root 1 0 0 14:52 ? 00:00:00 init [2]
root 3 1 0 14:52 ? 00:00:00 [ksoftirqd/0]
root 143 6 0 14:52 ? 00:00:00 [bdflush]
root 145 6 0 14:52 ? 00:00:00 [kswapd0]
root 357 6 0 14:52 ? 00:00:01 [kjournald]
root 477 1 0 14:52 ? 00:00:00 udevd --daemon
root 719 6 0 14:52 ? 00:00:00 [khubd]
```

Varios *daemons* de sistema, como kswapd *daemon*, que controla el intercambio de páginas con memoria virtual. Manejo de *buffers* del sistema (bdflush). Manejo de *journal* de *filesystem* (kjournald), manejo de USB (khubd). O el demonio de udev que controla la conexión en caliente de dispositivos. En general, no siempre los *daemons* suelen identificarse por una d final, y si llevan un k inicial, normalmente son hilos (*threads*) internos del *kernel*.

```
root    1567 1 0 14:52 ? 00:00:00 dhclient -e -pf ...
root    1653 1 0 14:52 ? 00:00:00 /sbin/portmap
root    1829 1 0 14:52 ? 00:00:00 /sbin/syslogd
root    1839 1 0 14:52 ? 00:00:00 /sbin/klogd -x
root    1983 1 0 14:52 ? 00:00:09 /usr/sbin/cupsd
root    2178 1 0 14:53 ? 00:00:00 /usr/sbin/inetd
```

Tenemos dhclient, esto indica que esta máquina es cliente de un servidor DHCP, para obtener su IP. Syslogd, un *daemon* que envía mensajes al log. El *daemon* de cups, como vimos, relacionado con el sistema de impresión. E inetd, que, como

veremos en la parte de redes, es una especie de “superservidor” o intermediario de otros *daemons* relacionados con servicios de red.

```
root     2154 1 0 14:53 ?          00:00:00 /usr/sbin/rpc.mountd
root     2241 1 0 14:53 ?          00:00:00 /usr/sbin/sshd
root     2257 1 0 14:53 ?          00:00:00 /usr/bin/xfs -daemon
root            2573 1 0 14:53 ?          00:00:00 /usr/sbin/atd
root     2580 1 0 14:53 ?          00:00:00 /usr/sbin/cron
root     2675 1 0 14:53 ?          00:00:00 /usr/sbin/apache
www-data        2684 2675 0 14:53 ?       00:00:00 /usr/sbin/apache
www-data        2685 2675 0 14:53 ? 00:00:00 /usr/sbin/apache
```

También está sshd, servidor de acceso remoto seguro (una versión mejorada que permite servicios compatibles con telnet y ftp). xfs es el servidor de fuentes (tipos de letra) de X Window. Los comandos atd y cron sirven para manejar tareas programadas en un momento determinado. Apache es el servidor web, que puede tener varios hilos activos (*threads*) para atender diferentes peticiones.

```
root 2499 2493 0 14:53 ?            00:00:00 /usr/sbin/gdm
root 2502 2499 4 14:53 tty7         00:09:18 /usr/bin/X :0 -dpi 96 ...
root 2848 1 0 14:53 tty2            00:00:00 /sbin/getty 38400 tty2
root 2849 1 0 14:53 tty3            00:00:00 /sbin/getty 38400 tty3
root 3941 2847 0 14:57 tty1         00:00:00 -bash
root 16453 12970 0 18:10 pts/2      00:00:00 ps -ef
```

Gdm es el login gráfico del sistema de escritorio Gnome (la entrada que nos pide el *login* y contraseña), los procesos getty son los que gestionan los terminales virtuales de texto (los que podemos ver con las teclas Alt+Fx (o Ctrl+Alt+Fx si estamos en modo gráfico). X es el proceso de servidor gráfico de X Window System, imprescindible para que se ejecute cualquier entorno de escritorio por encima de él. Un shell abierto (bash), y finalmente el proceso que hemos generado al pedir este *ps* desde la línea de comandos.

El comando ps tiene muchas opciones de línea de comandos para ajustar la información que queremos de cada proceso, ya sea tiempo que hace que se ejecuta, porcentaje de CPU usado, memoria utilizada... (ver man de ps). Otro comando muy interesante es top, que hace lo mismo que ps pero de forma dinámica, o sea, se actualiza con cierto intervalo, podemos clasificar los procesos por uso de CPU, de memoria, y también nos da información del estado de la memoria global.

Otros comandos útiles para uso de recursos son free y vmstat, que nos dan información sobre la memoria utilizada y el sistema de memoria virtual:

Nota

Ver man de los comandos para interpretar las salidas.

```
# free
         total used free shared buffers cached
Mem:     767736 745232 22504 0 89564 457612
-/+ buffers/cache: 198056 569680
Swap:    618492 1732 616760

# vmstat
procs -----------memory---------- ---swap-- -----io-- --system-- ----cpu----
 r b swpd free buff cache si so bi bo in cs us sy id wa
 1 0 1732 22444 89584 457640 0 0 68 137 291 418 7 1 85 7
```

En el comando free también se puede observar el tamaño del swap presente, aproximadamente de unas 600 MB, que de momento no se está usando intensamente por tener suficiente espacio de memoria física, todavía quedan unas 22 MB libres (lo que indica una alta utilización de la memoria física, y un uso de swap próximamente). El espacio de memoria y el swap (a partir de los kernels 2.4) son aditivos para componer el total de memoria del sistema, haciendo en este caso un total de 1,4 GB de memoria disponible. Esto puede parecer mucho, pero dependerá de las aplicaciones que se estén ejecutando.

Actividades

1) El espacio de swap permite complementar la memoria física para disponer de mayor memoria virtual. Dependiendo de los tamaños de memoria física y swap, ¿puede llegar a agotarse la memoria? ¿Podemos solucionarlo de otro modo, que no sea añadiendo más memoria física?

2) Supongamos que tenemos un sistema con dos particiones Linux, una / y la otra de swap. ¿Cómo solucionaríamos el caso de que las cuentas de los usuarios agotasen el espacio de disco? Y en el caso de tener una partición /home aislada que se estuviera agotando, ¿cómo lo solucionaríamos?

3) Instalad el sistema de impresión CUPS, definir nuestra impresora para que funcione con CUPS y probar la administración vía interfaz web. Tal como está el sistema, ¿sería recomendable modificar de algún modo la configuración que trae por defecto CUPS? ¿Por qué?

4) Examinad la configuración por defecto que traiga el sistema GNU/Linux de trabajos no interactivos por cron. ¿Qué trabajos y cuándo se están realizando? ¿Alguna idea para nuevos trabajos que haya que añadir?

5) Reproducid el análisis del taller (más los otros apartados de la unidad) sobre la máquina que se disponga, ¿se observan en el sistema algunos errores o situaciones anómalas? En tal caso, ¿cómo las corregimos?

Otras fuentes de referencia e información

[Wm02] [Fri02] [Smi02] Libros de administración de GNU/Linux y UNIX, donde se comentan de forma amplia aspectos de administración local y gestión de sistemas de impresión.

[Gt] Se puede encontrar información actualizada de los sistemas de impresión y su configuración, así como detalles de algunas impresoras. Para detalles concretos de modelos de impresora y controladores, podemos dirigirnos a http://www.linuxprinting.org/.

[Hin][Koe] Encontramos información sobre los diferentes sistemas de ficheros disponibles y los esquemas de creación de particiones para la instalación del sistema.